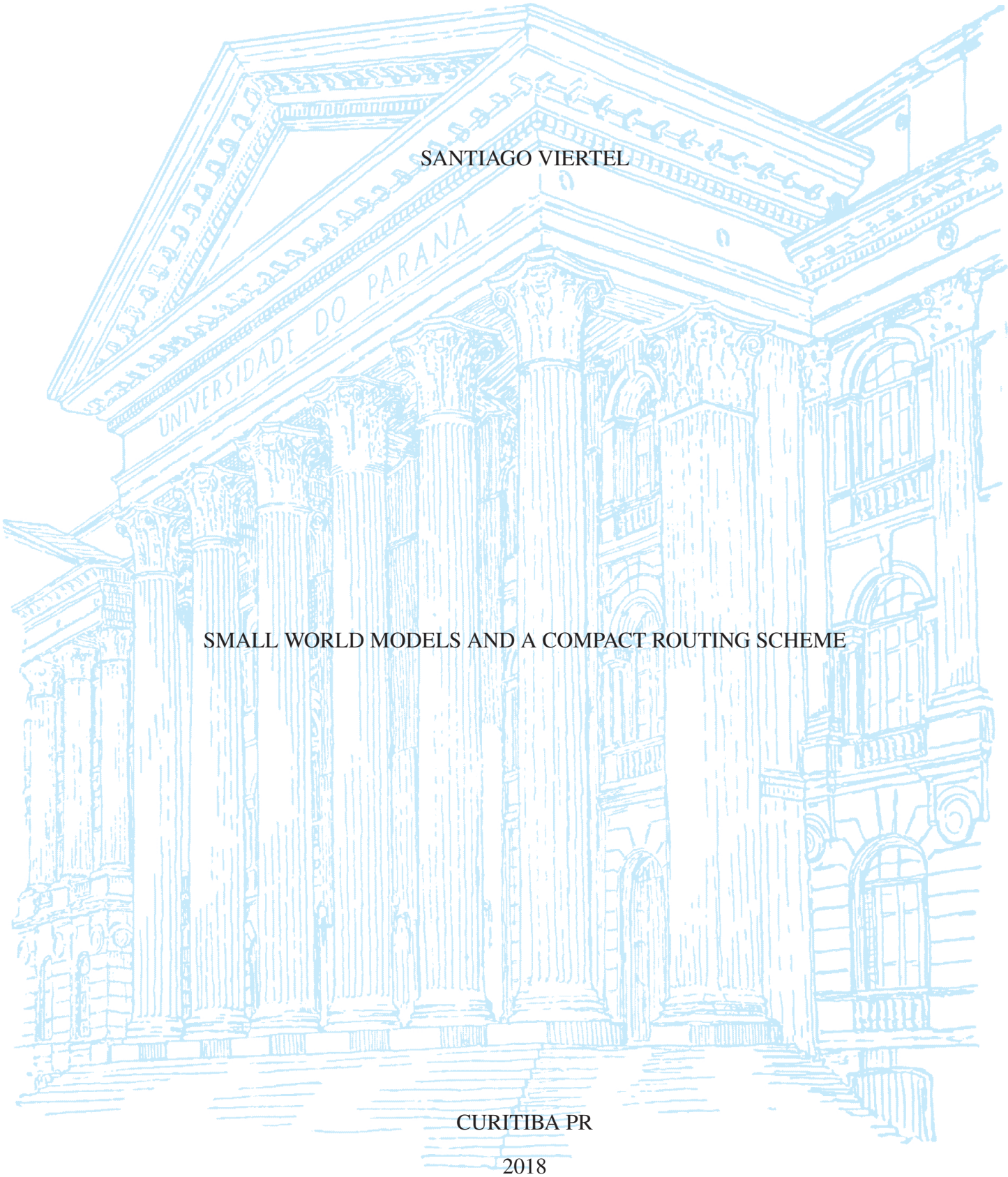
UNIVERSIDADE FEDERAL DO PARANÁ

SANTIAGO VIERTEL



SMALL WORLD MODELS AND A COMPACT ROUTING SCHEME

CURITIBA PR

2018

SANTIAGO VIERTEL

SMALL WORLD MODELS AND A COMPACT ROUTING SCHEME

Tese apresentada como requisito parcial à obtenção do grau de Doutor em Ciência da Computação no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Prof. Dr. André Luís Vignatti.

CURITIBA PR

2018

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

V665s Viertel, Santiago
Small world models and a compact routing scheme / Santiago Viertel. – Curitiba, 2018.

Tese - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática, 2018.

Orientador: André Luís Vignatti .

1. Redes locais de computadores. 2. Roteadores (Redes de computadores). 3. Compartilhamento de arquivos de computador. 4. Estrutura de dados (Computação). 5. Algorítmos computacionais. I. Universidade Federal do Paraná. II. Vignatti, André Luís. III. Título.

CDD: 004.68

Bibliotecário: Elias Barbosa da Silva CRB-9/1894

MINISTÉRIO DA EDUCAÇÃO
SETOR SETOR DE CIENCIAS EXATAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO INFORMÁTICA

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da tese de Doutorado de **SANTIAGO VIERTEL** intitulada: **Small World Models and a Compact Routing Scheme.**, após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de doutor está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 17 de Outubro de 2018.

ANDRÉ LUÍS VIGNATTI
Presidente da Banca Examinadora

ANDRÉ LUIZ PIRES GUEDES
Avaliador Interno

CLAUDIO CESAR DE SA
Avaliador Externo

SHEILA MORAIS DE ALMEIDA
Avaliador Externo

RENATO JOSÉ DA SILVA CARMO
Avaliador Interno

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
Setor de Ciências Exatas
Coordenação de Pós-Graduação em Informática

For my family and friends.

ACKNOWLEDGEMENTS

I would like to thank God for completing this stage of my life and my family for supporting me all the time. I especially thank my parents Ana Maria Viertel and Osni Viertel, who always have inspired me to achieve my goals and trusted my potential. I thank my girlfriend Elisangela Alves da Rocha for her companionship, patience and, mainly, for her affection. I thank my brother Roger Viertel and my sister Giuliana Viertel Bernardo, who supported me during the doctorate. Thank you very much, I love you all. I also thank all members of the Viertel and Reinert families.

I would like to thank my friends for the fun moments. I thank my advisor André Luís Vignatti for his teaching and for helping me to wisely guide my research. Besides being my advisor, he is also a great friend. I thank the reviewers André Luiz Pires Guedes, Renato José da Silva Carmo, Sheila Morais de Almeida and Claudio Cesar de Sá. They dedicated their time to provide me important considerations in the qualifying examination and in the defense of the thesis. I thank Professor Claudio Cesar de Sá for indicating me to the graduate program. I thank Professor André Luiz Pires Guedes for heartily welcoming me at the university.

I would like to thank all members of the Algorithms Research Group (ARG). They were present in many times, helping me with technical issues through informal conversations in coffee breaks. I thank all members of the Graduate Program in Informatics (PPGInf) of the Federal University of Paraná (UFPR) for the opportunity to develop my research. I thank the Coordination for the Improvement of Higher Education Personnel (CAPES) for the funding (Proc. 1431510/2014-9).

RESUMO

Os modelos geradores de redes de mundo pequeno são alternativas aleatorizadas de construção de redes onde os nós estão interligados por caminhos curtos. Nesse contexto, um caminho curto tem o comprimento dado por uma função logarítmica no tamanho da rede. Aplicações em redes *peer-to-peer*, de sensores sem fio e implementação de redes em *chip* são alguns exemplos computacionais que utilizam redes geradas por modelos matemáticos de mundo pequeno. Esse trabalho apresenta o modelo de redes de mundo pequeno toroidal não direcionado (`UTSW`). O modelo gera uma grade bidimensional base para modelar o agrupamento dos nós, e conexões aleatórias para modelar caminhos curtos. A presença de caminhos curtos não implica a existência de um algoritmo de roteamento de mensagens que os encontra. Porém, existe um algoritmo de roteamento guloso na literatura que encontra caminhos curtos em redes `UTSW`, sendo que um caminho curto tem o comprimento dado por uma função polilogarítmica no tamanho da rede. O modelo de mundo pequeno octaédrico (`OSW`) também é apresentado. Ele gera um grafo planar base sobre o octaedro, que é provado ser similar à geração sobre esferas. Assim, ele simula pessoas e vizinhanças na superfície do planeta. O modelo `OSW` também gera conexões aleatórias para modelar caminhos curtos. Um algoritmo de roteamento que encaminha mensagens por caminhos curtos também é definido para esse modelo. Ambos os algoritmos de roteamento utilizam o paradigma guloso e decidem para qual vizinho encaminhar uma mensagem com somente a informação na tabela de roteamento do nó. Esses algoritmos requerem pouca memória por nó, encontram caminhos curtos e executam em tempo constante, nos dois modelos apresentados. Por isso, eles são boas opções em roteamento, além do roteamento por caminhos mínimos. No entanto, ambos necessitam de informações de posicionamento geradas pelo modelo matemático. No caso do modelo `UTSW`, cada nó possui um par ordenado de números naturais que o posiciona na grade bidimensional. Encontrar essas posições quando elas são desconhecidas é desafiador, pois as conexões geradas aleatoriamente dificultam a identificação da grade. Porém, a conexão aleatória de um nó quebra o padrão de grade em sua vizinhança com baixa probabilidade. Essa propriedade topológica das redes `UTSW` motivou o projeto de um algoritmo de rotulação que encontra as posições na grade com alta probabilidade. Ele realiza uma busca local em cada nó e remove a conexão aleatória, se ela for identificada. Após a varredura, o algoritmo realiza uma busca em largura global na rede resultante, posicionando cada nó com base nas posições já encontradas de seus vizinhos. Um esquema de roteamento compacto para grafos `UTSW` é então apresentado. Ele é composto por um algoritmo de pré-processamento que utiliza o algoritmo de rotulação para gerar estruturas de dados de tamanho logarítmico para todos os nós e que executa em tempo linear no tamanho da rede. Com isso, o algoritmo de roteamento guloso encaminha mensagens utilizando as estruturas de dados geradas pelo algoritmo de pré-processamento.

Palavras-chave: redes de mundo pequeno. modelos geradores. roteamento guloso. algoritmos de rotulação. esquemas de roteamento compactos.

ABSTRACT

Small world networks generative models are randomized alternatives of network building where nodes are interconnected by small paths. In this context, a small path has the length given by a logarithmic function in the size of the network. Applications in peer-to-peer networks, wireless sensors and networks on chip are some computational examples that use networks generated by small world mathematical models. This work presents the undirected toroidal small world networks model (UTSW). The model generates a base two-dimensional grid to model the clustering of the nodes, and random connections to model small paths. The presence of small paths does not imply in the existence of a message routing algorithm that finds them. However, there is a greedy routing algorithm in the literature that finds small paths in UTSW networks, where a small path has the length given by a polylogarithmic function in the size of the network. The octahedral small world model (OSW) is also presented. It generates a base planar graph on the octahedron, which is proved to be similar to the generating on spheres. Then, it simulates people and neighborhoods on the surface of the planet. The OSW model also generates random connections to model small paths. A routing algorithm that routes messages through small paths is also defined for this model. Both routing algorithms use the greedy paradigm and decide to which neighbor to forward a message with only the information in the routing table of the node. These algorithms require small amount of memory per node, find small paths and execute in constant time, in both presented models. So, they are good options in routing, besides the routing thought shortest paths. However, both require positioning information generated by the mathematical model. In the case of UTSW model, each node has an ordered pair of natural numbers that positions it on the two-dimensional lattice. Finding these positions when they are unknown is challenging because the randomly generated connections make the grid identification difficult. However, the random connection of a node breaks the lattice pattern in its neighborhood with small probability. This topological property of UTSW networks motivated the design of a labeling algorithm that finds the positions in the lattice with high probability. It executes a local search on each node and removes the random connection if it is identified. After the sweeping, the algorithm executes a global breadth-first search on the resulting network, positioning each node based on the already found positions of its neighbors. A compact routing scheme for UTSW graphs is then presented. It consists of a preprocessing algorithm that uses the labeling algorithm to generate logarithmic size data structures for all nodes and executes in linear time on the size of the network. Thus, the greedy routing algorithm routes messages using the data structures generated by the preprocessing algorithm.

Keywords: small world networks. generative models. greedy routing. labeling algorithms. compact routing schemes.

LIST OF FIGURES

2.1 A metric embedding f from (V, d) to $(\mathbb{R}^2, \ell_2)$. 16
2.2 A two-dimensional square grid of size $n = 6$. 17
2.3 Lattice distance between two vertices $u, v \in V$. 18
2.4 Two two-dimensional square tori.. 18
2.5 Torus distance between two vertices $u, v \in V$. 19
2.6 Routing tables sizes of compact routing schemes. 22
2.7 Routing table of a vertex $u \in V$. 22
2.8 Graphs with $n = 20$, $p = 4$ and distinct values of β (Watts and Strogatz, 1998). . 26

3.1 A base graph (A) and the neighborhood of a vertex (B) (Kleinberg, 2000a). . . . 29
3.2 An execution of Thorup and Zwick's routing algorithm.. 35

4.1 Vertices at a distance $i = 3 < n/2$ from u, with $n = 7$. 41
4.2 Graph with more than one spanning torus. 43

5.1 A two-octahedral graph. 51
5.2 The underlying cycles surrounding an axis. 52
5.3 Distance between the projections of u and v, u' and v' respectively, on a sphere with radius $r > 0$. 53
5.4 The upper bound function of $r > 0$. 53
5.5 Right triangles defined by u, v and O_n.. 54
5.6 Sets of vertices at the distances one, two and three from the central vertex. 55
5.7 Some vertices in B_{tj} with distance of at most $2^j \leq n$ from $t \in V$ in an octahedron folding. 55

6.1 Four four-cycles that constitutes a lattice pattern. 65
6.2 Possible references to label the cross.. 71

CONTENTS

1	**Introduction**	**10**
1.1	The Problem	11
1.2	Contributions	12
1.3	Organization.	12
2	**Theoretical Foundations.**	**14**
2.1	Metric Spaces	14
2.1.1	Connected Graphs and Metric Spaces	15
2.1.2	Metric Embeddings	15
2.2	Vertex Labellings	16
2.2.1	Two-Dimensional Square Grids.	17
2.2.2	Two-Dimensional Square Tori	18
2.3	Compact Routing Schemes	21
2.3.1	Routing Tables	22
2.3.2	Labels and Headers	23
2.3.3	Preprocessing Algorithms.	24
2.3.4	Routing Algorithms	24
2.3.5	Efficiency Measurements	24
2.4	Random Small World Graphs	25
2.5	Small and High Probabilities	27
3	**Literature Review**	**28**
3.1	Routing in Small World Models	28
3.2	Routing with Compact Structures and Bounded Stretch	33
3.3	Other Related Works	37
4	**Small World on the Torus**	**40**
4.1	Undirected Toroidal Small World Model	40
4.2	Toroidal Small World Labeling Problem	42
4.3	Cycles of Size Four Outside the Torus	43
5	**Small World on the Octahedron and Spheres**	**50**
5.1	Octahedrons and Graphs	50
5.1.1	The n-Octahedral Graph and Spheres.	52
5.2	Octahedral Small World Model	53
5.3	Three-Cycles not in the n-Octahedral Graph	56

6 A Compact Routing Scheme 61
6.1 Bounded Search 62
6.2 Lattice Pattern and Detection 64
6.3 Removing Long-Range Edges 66
6.4 Labeling the Reference System 68
6.5 Labeling Arbitrary Crosses 71
6.6 Main Algorithm 73
7 Conclusion 75
7.1 Future Works 76
REFERENCES 78

1 Introduction

This work presents two random small world networks generative models and a solution to the problem of compact routing in small world networks generated by one of them. Message routing in computer networks is a classic problem in computer science. Limitations in processing and storage of technological devices are issues that make this problem defiant. The literature possess analytical results on message routing in computer networks (Cowen, 1999; Thorup and Zwick, 2001; Bringmann et al., 2017). Compact routing schemes portray some of those results. A **routing scheme** provides a complete mechanism for routing messages in a network. It consists of (i) a **preprocessing algorithm** that generates, for each node of the network, data structures storing routing information and (ii) a **routing algorithm** that routes messages and executes in all nodes of the network. In this context, the preprocessing algorithm takes as the input a graph that represents a computer network and outputs, for each node, its address in the network, its routing table and other auxiliary structures. The routing algorithm takes as the input a message that reaches the local node, the node address, routing table and other auxiliary structures, and it forwards the message to the neighbor which it considered closest to some target node.

A routing scheme is **compact** if each node uses sublinear storage space in the number of nodes of the network, that is, $o(n)$ space for a network with n nodes. A compact routing scheme may not route messages along shortest paths because it only allows structures with sublinear sizes. There are schemes with different trade-offs between the sizes of the data structures per vertex and the sizes of the paths. For example, a scheme may decrease the paths sizes at the cost of a small increase in the structures sizes. The worst case of number of bits required for storage per vertex measures the effectiveness of routing schemes. Moreover, the average number of bits for storage is also considered. Some works also measure the effectiveness through the **stretch factor** (Cowen, 1999; Abraham et al., 2006a; Chen et al., 2009), which is the maximal ratio, over all pairs of vertices, between the length of the path obtained by the routing scheme and the length of a shortest path. Others measure through the **expected paths length** performed by the routing scheme (Kleinberg, 2000a; Zeng et al., 2005; Liu et al., 2009). These concepts are formally defined in Section 2.3.

There are "universal" compact routing schemes in the sense that they are optimal solutions for the size of the data structures per vertex in relation to the size of the paths for any undirected graph topology (Cowen, 1999; Thorup and Zwick, 2001). For example, Thorup and Zwick (2001) present a compact routing scheme for general graphs with stretch factor 3 that requires almost optimal $\mathcal{O}\left((n \cdot \log n)^{1/2}\right)$ bits per vertex for this stretch factor, where n is the number of vertices. However, better adjustments are attainable when the scheme is specialized for a given family of graphs. For example, there are compact routing schemes specialized in trees (Thorup and Zwick, 2001; Fraigniaud and Gavoille, 2001), in graphs that induce metric spaces with low doubling dimension (Abraham et al., 2006a; Konjevod et al., 2007b) and in planar graphs (Gavoille and Hanusse, 1999; Lu, 2002). There are also schemes specialized in mathematical models of random power-law graphs (Chen et al., 2009).

There are mathematical models that deal with random small world graphs. **Small world graphs** have clustering of vertices and short paths that interconnect most pairs of vertices. Small world models have applications in networks of the electronic mail messages exchanging (Adamic and Adar, 2005), friendship (Liben-Nowell et al., 2005), Internet domain (Krioukov et al., 2004), peer-to-peer (Manku et al., 2003), mobile ad hoc (Liu and Wu, 2006), wireless sensors (Liu et al., 2009) and on chip (Shamim et al., 2017). This thesis also presents two small world generative models, defined in Chapters 4 and 5. Both chapters also prove some topological properties of the graphs that the models generate through probabilistic analyzes.

Kleinberg (2000a) presents a small world model and a greedy routing algorithm for his model. The model generates a $n \times n$ lattice of vertices $V = \{1, 2, ..., n\} \times \{1, 2, ..., n\}$. Each vertex $(i, j) \in V$ has a **position in the lattice** (i, j) such that $1 \leq i \leq n$ and $1 \leq j \leq n$. He defines the **lattice distance** between two vertices $(i, j), (k, l) \in V$ as $d((i, j), (k, l)) = |k - i| + |l - j|$. The model has three parameters, $p \geq 1$, $q \geq 0$ and $r \geq 0$. It links each vertex with directed edges to the vertices within lattice distance p, and for each $u \in V$, it generates q directed **long-range edges** to $v \in V \setminus \{u\}$ with probability proportional to $d^{-r}(u, v)$. The probability of each v is multiplied by the **normalizing factor** $\left(\sum_{w \in V \setminus \{u\}} d^{-r}(u, w)\right)^{-1}$. Kleinberg also presents a greedy routing algorithm that forwards a received message to the neighbor closest to the target in the lattice distance. He proves that, for $p = q = 1$ and $r = 2$, this routing algorithm finds paths with $\mathcal{O}\left(\log^2 n\right)$ expected length, which is considered small. Some works present modifications on the Kleinberg's model and on the greedy routing algorithm to obtain improvements or trade-offs between storage required in bits per vertex and paths lengths (Manku et al., 2004; Fraigniaud et al., 2006; Zeng and Hsu, 2006).

1.1 The Problem

Kleinberg (2006) presents another problem in compact routing in random small world graphs. Kleinberg's greedy routing algorithm requires that each vertex stores the position in the lattice of all its neighbors. The problem is to compute the position in the lattice (labels) of all vertices when the positions are unknown, taking as the input only the graph $G = (V, E)$ generated by the small world model. Note that each vertex $(i, j) \in V$ represents a position in the lattice by definition, however this thesis considers the problem where the vertices do not possess any positioning information at all. The difficulty arises because the long-range edges generation "hides" the underlying lattice and makes this problem not easy to solve.

Sandberg (2006) claims that in some applications, such as in peer-to-peer, a vertex may not store the positions of its neighbors. He then addresses the problem experimentally, by using a Markov chain Monte Carlo algorithm that estimates the positions of all vertices. He claims that the Kleinberg's greedy routing algorithm finds small paths with a good estimate for the positions in the lattice. The algorithm computes the positions of all vertices based on the hypothesis that the graph was generated by the Kleinberg's model over a **torus**. In this context, a torus is similar to the grid built by the Kleinberg's model with $p = 1$ and with the "borders linked".

The Section 4.2 presents the formal definition of this problem. Chapter 6 defines a labeling algorithm that find the positions in the lattice of almost all vertices of graphs generated by the model presented in the Section 4.1. Besides, the same chapter also defines a compact routing scheme that uses the presented labeling algorithm.

1.2 Contributions

Roughly speaking, the contributions of this thesis is twofold: we present two models for small world graphs, together with properties based on that models; and also a compact routing scheme for one of the presented models.

The first model is based on the Kleinberg's model and called **undirected toroidal small world** (UTSW). It generates random graphs over a base torus rather than over a lattice where each vertex has edges to $p \geq 1$ closest neighbors in the lattice. Moreover, UTSW model generates only undirected edges, opposed to the Kleinberg's model. So, if a vertex $u \in V$ creates a long-range edge to $v \in V \setminus \{u\}$, then v has a long-range edge to u. The normalizing factor in UTSW model is $\Theta\left(\log^{-1} n\right)$, where n is a parameter of the model that defines the size of the torus. We also bound the probability of existence of four-cycles composed by edges not in the base torus. This event occurs with small probability of $\mathcal{O}\left(\log^{-1} n\right)$. The small probability allows the design of a labeling algorithm that executes in expected linear time. Then, we define the labeling problem for UTSW graphs, present the algorithm that labels almost all vertices of UTSW graphs and define a compact routing scheme for UTSW model.

The second model is the **octahedral small world** (OSW), that generates the graph over a sphere homeomorphic geometry. The vertices lie on the surface of an octahedron and are connected with nearby neighbors, generating the base graph. This model also generates three-cycles not in the base graph with small probability of $\mathcal{O}\left(\log^{-1} n\right)$, where n is a parameter of the model that defines the size of the octahedron.

This PhD thesis resulted in two papers, one titled "**Labeling Algorithm and Compact Routing Scheme for a Small World Network Model**" (Viertel and Vignatti, 2018a) and the other "**Small World Model based on a Sphere Homeomorphic Geometry**" (Viertel and Vignatti, 2018b).

The first of them defines the UTSW model and the problem of labeling vertices of UTSW graphs with the positions in the lattice. It also proves the upper bound for the probability of existence four-cycles outside the torus and presents the labeling algorithm and compact routing scheme for UTSW graphs. It was submitted for publishing to "Theoretical Computer Science" journal and has its results described in Chapters 4 and 6.

The second paper defines the base graph generated on the octahedron, the relation between the base graph and spheres, the OSW model and the bounds for the probability of existence three-cycles outside the base octahedral graph and for the expected number of three-cycles of OSW graphs. It was submitted for publishing to "Information Processing Letters" journal and has its results described in Chapter 5.

1.3 Organization

Each chapter describes its own organization at the beginning. Some notation repeat among the chapters but sometimes having different meanings. They are used in different papers written during the project (Viertel and Vignatti, 2018a,b) and do not change in this work. The organization of the sections allows these notation being maintained without loss of coherence between the concepts. Besides that, many of the notation are mnemonic, which facilitates the association with the local concepts of each chapter. Therefore, the repeated notation are locally redefined as necessary. Some definitions are also locally redefined throughout the thesis for convenience. In general, Chapters 3 to 6 refer to concepts presented in Chapter 2. Specially, Chapter 6 refer many times to Chapter 4 because there are many conceptual dependencies between them. There are

three logarithm notations in the entire thesis: $\log$, $\log_2$ and $\ln$. The logarithm has the base 10 when not specified and $\ln$ is the natural logarithm.

The Chapter 2 presents concepts about metric spaces and embeddings, two specific vertex labellings for grids and tori, compact routing schemes, random small world graphs generation and small and high probabilities. Chapter 3 presents related works about small world graph models, routing algorithms for the models, compact routing schemes for general graphs, compact routing schemes specialized for some families of graphs and some more related works. Chapter 4 presents the definition of the `UTSW` model, an analysis of its normalizing factor, the formal definition of the **toroidal small world labeling problem** and a sequence of results used to bound the probability of existence of four-cycles that are not in the base torus. Chapter 5 presents the generation of a base graph over the octahedron, how the base graph is related with spheres, the definition of the `OSW` model, the definition of a greedy routing algorithm and a sequence of results used to bound the number of three-cycles that are not in the base octahedron. Chapter 6 presents all procedures of the labeling algorithm for `UTSW` graphs, their respective probabilistic analyzes and finishes defining a compact routing scheme for `UTSW` graphs. Finally, Chapter 7 presents the conclusion of the thesis and future works.

2 Theoretical Foundations

This chapter presents concepts used throughout the thesis. Readers familiar with the concepts should read quickly, focusing on the notations. All notations used more than once are locally redefined in each section as necessary. The proofs of the Theorems 2 and 10 are part of the results obtained in thesis.

2.1 Metric Spaces

A metric function maps a pair of elements of a set to a real number. It has the three properties, non-negativeness, symmetry and triangle inequality. Non-negativeness implies in mapping of each pair in a non-negative number, such that the number is 0 if, and only if, both elements of the pair are the same. Symmetry implies in the two permutations of each pair mapped in the same number. Triangle inequality implies in the mapping of each pair (u, v) not to be greater than the sum of the mappings of the pairs (u, w) and (w, v). The definition of metric function follows.

Definition 1. Let V be a set. A function $d : V \times V \to \mathbb{R}$ is a **metric** if, and only if, for each $u, v, w \in V$:

- $d(u, v) \geq 0$, with equality if, and only if, $u = v$ (**non-negative property**);
- $d(u, v) = d(v, u)$ (**symmetric property**);
- $d(u, v) \leq d(u, w) + d(w, v)$ (**triangle inequality property**).

The term "**distance**" has the same meaning of metric in this thesis. A **metric space** is a par (V, d), where d is a metric function on the set V. Metric spaces can mathematically models real-world phenomena. For example, a metric space can be defined by a set of cities and a metric function that assigns a non-negative real number related to the distance in miles between each pair of cities. Another example is the metric space defined by the three-dimensional real space and the Euclidean distance, which could model a computational physical simulation.

For example, the metric space defined by the three-dimensional real space and the Euclidean distance is denoted $(\mathbb{R}^3, \ell_2)$. The notation S^k represents in this work the k-dimensional Cartesian product of the set S, where $k \in \mathbb{N}_*$. Moreover, $\mathbb{N}_*$ represents the set of the natural numbers without 0, that is, $\mathbb{N}_* = \{1, 2, ...\}$. For the given example, the metric ℓ_2 corresponds to the distance function $d(u, v) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + (u_3 - v_3)^2}$ for each $u, v \in \mathbb{R}^3$, where $w = (w_1, w_2, w_3)$. More generally, for every $p \geq 1$, the **length** of u in the metric space $\left(\mathbb{R}^k, \ell_p\right)$ is the function

$$||u||_p = \left(\sum_{i=1}^{k} |u_i|^p\right)^{1/p},$$

where $k \in \mathbb{N}_*$ is the dimension of the space, $u \in \mathbb{R}^k$ is an element of the space of dimension k and u_i is the value of the coordinate i of u. The metric ℓ_p, also referred as p-**norm**, is the function $d(u, v) = ||u - v||_p$ between each pair $u, v \in \mathbb{R}^k$. In the example of the metric space $(\mathbb{R}^3, \ell_2)$, the dimension $k = 3$, the norm parameter $p = 2$ and $d(u, v) = \left(\sum_{i=1}^{3} |u_i - v_i|^2\right)^{1/2}$. All norms have the properties of Definition 1. Further details about metric spaces and norms can be found in the book of Searcóid (2007).

2.1.1 Connected Graphs and Metric Spaces

Let $G = (V, E)$ be a connected and undirected graph without weights and d be a metric function that maps each pair $u, v \in V$ to the **number of edges of a minimum path** from u to v in G. Theorem 2 proves that the pair (V, d) is a metric space defined by G.

Theorem 2. *Let $G = (V, E)$ be a connected and undirected graph and d be the number of edges of a minimum path between each $u, v \in V$ in G. Then G defines the metric space (V, d).*

Proof. Function d must have all properties of Definition 1. It has the **non-negativity** property because there is no path with negative length in G and because $d(u, u) = 0$ for all $u \in V$. It also has the **symmetry** property because a minimum path in G from the vertex $u \in V$ to the vertex $v \in V$ has the same number of edges of a minimum path from v to u, for all $u, v \in V$.

The proof that d has the **triangle inequality** property follows by contradiction. Suppose that there exists three vertices $u, v, w \in V$ such that $d(u, v) > d(u, w) + d(w, v)$ in G. Let p_{uv} be a minimum path from u to v in G with $d(u, v)$ edges, p_{uw} be a minimum path from u to w in G with $d(u, w)$ edges and p_{wv} be a minimum path from w to v in G with $d(w, v)$ edges. Let p_{uwv} be the concatenation of p_{uw} and p_{wv} with $d(u, w) + d(w, v)$ edges. As the inequality $d(u, v) > d(u, w) + d(w, v)$ holds by assumption, so there exists a path p_{uwv} from u to v with fewer edges than the minimum path p_{uv} from u to v, which is a contradiction. Therefore, d also has the triangle inequality property and (V, d) is a metric space. □

The Theorem 2 can not be generalized for directed graphs because d may break the symmetry property. On the other hand, the proof is similar for connected and undirected graphs with positive weights in the edges and distance function defined by the **length of a minimum weighted path**. The works presented in the Section 3.2 assume that the input graph defines a metric space.

2.1.2 Metric Embeddings

A metric embedding function maps each element of a metric space to an element of another. Let (V, d) and (V', d') be two metric spaces, a **metric embedding** is a function $f : V \to V'$. For example, let (V, d) be the metric space defined by the undirected tree $T = (V, E)$. There exists a metric embedding function $f : V \to \mathbb{R}^{|V|-1}$ that relates (V, d) with the metric space $\left(\mathbb{R}^{|V|-1}, \ell_1\right)$. Viertel and Vignatti (2015) define this metric embedding function.

An isometric embedding function relates a metric space with another such that the distances in the first remain unchanged in the second, as Definition 3 shows. The metric embedding defined by Viertel and Vignatti is isometric and they also sketch the prove. Figure 2.1 shows another example of metric embedding. The metric embedding function f maps two elements of (V, d) to the respective two elements of $(\mathbb{R}^2, \ell_2)$. If f is isometric in this example, then $d(u, v) = \sqrt{(1-4)^2 + (6-2)^2} = 5$.

Definition 3. Let (V, d) and (V', d') be two metric spaces and $f : V \to V'$ be a metric embedding function, f is **isometric** if, and only if, $d(u, v) = d'(f(u), f(v))$ for each $u, v \in V$.

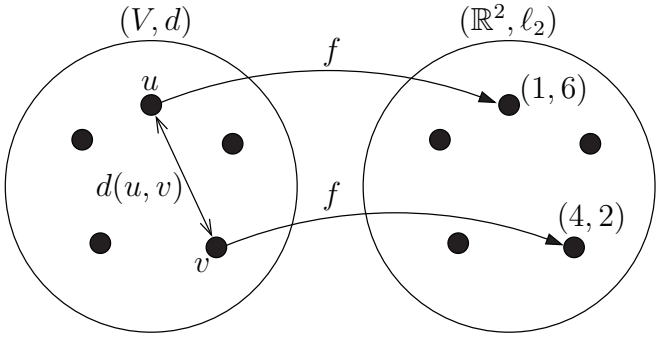


Figure 2.1: A metric embedding f from (V, d) to $\left(\mathbb{R}^2, \ell_2\right)$.

Let $G = (V, E)$ be an connected and undirected graph with positive weights in the edges that have the triangle inequality property, defining the metric space (V, d). Suppose that (V, d) is defined in the same way of the example of the Figure 2.1. That is, G defines (V, d), which has a mapping f to $(\mathbb{R}^2, \ell_2)$. Suppose that f is also isometric, then $d(u, v) = d'(f(u), f(v))$ for all $u, v \in V$. Hence, the Euclidean distance from $f(u)$ to $f(v)$ is equals to the length of a minimum weighted path from u to v in G. In this case, the distance from u to v in G is computed in **constant time** rather than, for example, in $\mathcal{O}(|E| + |V| \cdot \log |V|)$, demanded by Dijkstra algorithm implemented with Fibonacci heap (Cormen et al., 2009). Metric embeddings is an efficient and elegant mathematical tool for designing algorithms, allowing good solutions when used wisely (Williamson and Shmoys, 2010).

2.2 Vertex Labellings

Any information is represented by a k-tuple of elements in the set $\{0, 1\}$ with bounded dimension $k \in \mathbb{N}$. For example, the tuple $(1, 0, 0, 1, 1, 0)$ represents the integer number 38 because it is the binary notation of this number. Similarly, a tuple of dimension $k = 8$ represents a character in the ASCII table. The term "information" corresponds to a sequence of binary characters in this thesis, as Definition 4 shows. The notation $\{0, 1\}^*$ represents the infinite set of all tuples of all dimensions $k \in \mathbb{N}$ defined by the set $\{0, 1\}$.

Definition 4. An **information** is an element of $\{0, 1\}^*$.

The assignment of information to the vertices of a graph is defined by a labeling function, that maps the vertices to informations of dimension k, as Definition 5 shows. The notation $\{0, 1\}^k$ represents the finite set of all tuples of dimension $k \in \mathbb{N}$. Note that any sequence of informations of bounded dimension can be represented by a single k-tuple. For example, given a graph $G = (V, E)$ and a vertex $u \in V$, a labeling function l can map the vertex u to the pair of integer numbers $(10, 5)$. In this case, there is one information that encodes the number 10 and one more that encodes the number 5. Each of these two information can be represented by a sequence of 4 bits. Encoding each in binary notation and doing a concatenation, the mapping l from u to $(10, 5)$ is $l(u) = (1, 0, 1, 0, 0, 1, 0, 1)$. In this example, the tuple $l(u)$ is an information of dimension $k = 8$, where the first four coordinates represent the integer number 10 and the last four coordinates represent the integer number 5. Chapter 3 presents works that bound the storage used by algorithms by bounding the dimension k of informations. Next sections define labels as

pairs of natural numbers for convenience, however all labels are represented computationally as an information with bounded dimension.

Definition 5. A **vertex labeling** in a graph $G = (V, E)$ is a function $l : V \to \{0, 1\}^k$, where $k \in \mathbb{N}$.

2.2.1 Two-Dimensional Square Grids

A vertex labeling function may also be a metric embedding. For example, let $G = (V, E)$ be a connected and undirected graph that defines the metric space (V, d). It is possible to define a vertex labeling l in G that assigns pairs of natural numbers to each vertex $u \in V$ so that l is an isometric embedding from (V, d) to $(\mathbb{N}^2, \ell_1)$. In this context, each number can be represented by an information of dimension at most $\lfloor \log_2 x \rfloor + 1$, where x is the largest number among all pairs. Two-dimensional square grids, described in Definition 6, are a family of graphs where such embeddings do exist. The term "**lattice**" is used in this thesis to refer a Cartesian product $\mathbb{N}^k$ with a dimension $k \in \mathbb{N}_*$. Moreover, "**grid**" is used to refer a graph with topology described in Definition 6. Figure 2.2 shows a two-dimensional square grid of size $n = 6$.

Definition 6. The **two-dimensional square grid** of size $n \geq 1$ is the undirected graph $G = (V, E)$ with n^2 vertices arranged in a $n \times n$ lattice and E defined by all pairs of vertices adjacent in the lattice.

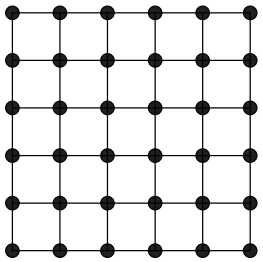


Figure 2.2: A two-dimensional square grid of size $n = 6$.

The two-dimensional Cartesian product of the undirected path of size n is an alternative definition for the two-dimensional square grid. The distance between a pair of vertices in a two-dimensional square grid is also computed through an isometric embedding l to $(\mathbb{N}^2, \ell_1)$. For example, let $G = (V, E)$ be a two-dimensional square grid of size 5 and (V, d) be the metric space defined by G. There exists at least one vertex labeling l that assigns pairs in $\{1, 2, ..., 5\}^2$ to the vertices of G, such that the number of edges of a minimum path $d(u, v)$ in G between each pair of vertices $u, v \in V$ is equals to the distance ℓ_1 between the labels $l(u)$ and $l(v)$. The Figure 2.3 shows G with edges omitted and the vertex labeling l, which is also an isometric embedding from (V, d) to $(\{1, 2, ..., 5\}^2, \ell_1)$, where $\{1, 2, ..., 5\}^2 \subset \mathbb{N}^2$. The norm ℓ_1 used in this context is also named lattice distance, as Definition 7 shows. Note that this definition has an abuse of notation, where l is actually an isometric embedding from (V, d) rather than G.

Definition 7. Let $G = (V, E)$ be a two-dimensional square grid of size $n \geq 1$ and l be a vertex labeling that is an isometric embedding from G to $(\{1, 2, ..., n\}^2, \ell_1)$, the **lattice distance** $d : V^2 \to \mathbb{N}$ is

$$d(u, v) = \sum_{i=1}^{2} |l(u)_i - l(v)_i|$$

for each $u, v \in V$, where $l(w)_i$ is the coordinate i of the pair/label $l(w)$ for each $w \in V$.

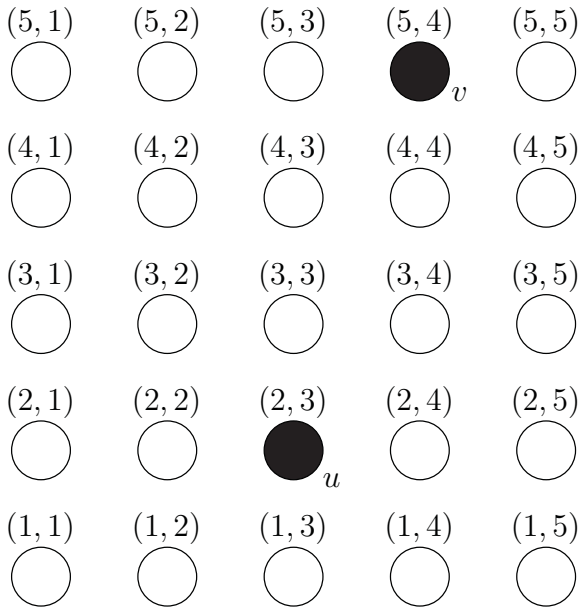
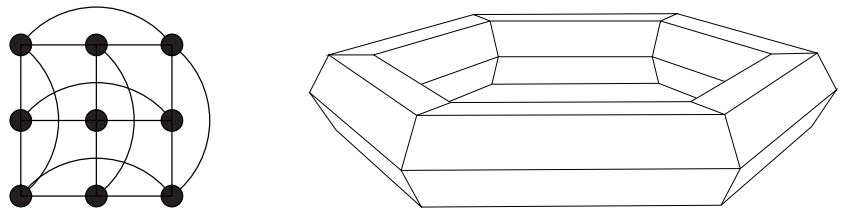


Figure 2.3: Lattice distance between two vertices $u, v \in V$.

2.2.2 Two-Dimensional Square Tori

Two-dimensional square tori are similar to two-dimensional square grids, however their "opposite boundaries" are connected. Figure 2.4(a) shows a two-dimensional square torus of size 3 and Figure 2.4(b) shows geometrically a two-dimensional square torus of size 6. Note that a two-dimensional square torus has size at least 3 because loops and parallel edges are not allowed. Definition 8 formalizes the two-dimensional square torus. There are more general definitions for toroidal graphs, however this suffices for this thesis.

Definition 8. The **two-dimensional square torus** of size $n \geq 3$ is the undirected graph G defined by the two-dimensional Cartesian product of the undirected cycle of size n.



(a) Torus of size 3. (b) Torus of size 6 geometrically viewed.

Figure 2.4: Two two-dimensional square tori.

Two-dimensional square tori have isometric embeddings to metric spaces defined by a specific metric function. Definition 9 formalizes the torus distance, where n is computed by $\sqrt{|V|}$. Figure 2.5 defines a vertex labeling l that is an isometric embedding. Note that the torus distance is similar to the lattice distance. However, it is the length of the shortest path between a path that does not pass through an edge that connects "vertices in the boundary of the lattice" and a path that passes through one of these edges.

Definition 9. Let $T = (V, E)$ be a two-dimensional square torus of size $n \geq 3$ and l be a vertex labeling that is an isometric embedding from T to $(\{1, 2, ..., n\}^2, d')$, where

$$d'(l(u), l(v)) = \sum_{i=1}^{2} \min\{|l(u)_i - l(v)_i|, n - |l(u)_i - l(v)_i|\} \text{ for all } u, v \in V$$

and $l(w)_i$ is the coordinate i of the pair/label $l(w)$ for each $w \in V$. The **torus distance** $d : V^2 \to \mathbb{N}$ is $d(u, v) = d'(l(u), l(v))$.

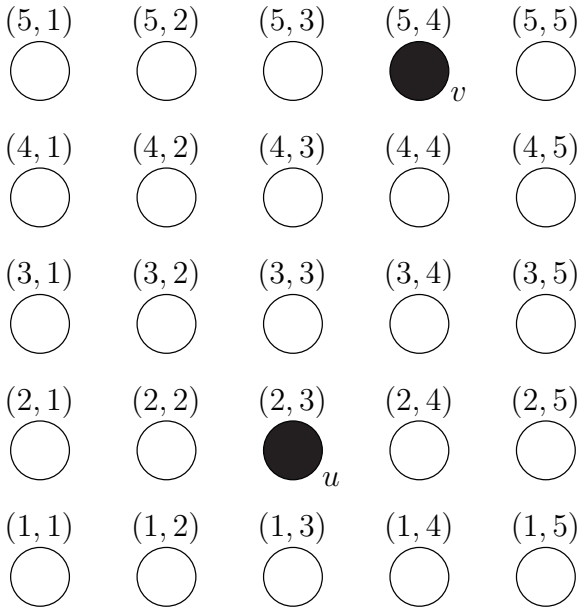


Figure 2.5: Torus distance between two vertices $u, v \in V$.

Figure 2.5 shows a vertex labeling l that is an isometric embedding to $(\{1, 2, ..., 5\}^2, d')$, where d' is in Definition 9. Note that the torus distance between pairs of vertices is at most the lattice distance between the same pairs, as Figures 2.3 and 2.5 show. As d is a metric function by Theorem 2 and $d(u, v) = d'(l(u), l(v))$ for all $u, v \in V$ by Definition 9, so d' is also a metric function. Even so, Theorem 10 proves the same result in another way and, consequently, $(\{1, 2, ..., n\}^2, d')$ is a metric space.

Theorem 10. *The function* $d'(a, b) = \sum_{i=1}^{2} \min\{|a_i - b_i|, n - |a_i - b_i|\}$ *is a metric function on* $\{1, 2, ..., n\}^2$*, where* $a, b \in \{1, 2, ..., n\}^2$*,* $a = (a_1, a_2)$ *and* $b = (b_1, b_2)$*.*

Proof. The function d' has the non-negative property of Definition 1, as proved next. The value $d'(a, b) \geq 0$ because $|a_i - b_i| \leq n - 1$ for all $a, b \in \{1, 2, ..., n\}^2$ and $i \in \{1, 2\}$. Besides, if $d'(a, b) = 0$, then $\min\{|a_i - b_i|, n - |a_i - b_i|\} = 0$ and $a_i = b_i$ for all $a, b \in \{1, 2, ..., n\}^2$ and $i \in \{1, 2\}$. Therefore, if $d'(a, b) = 0$, then $a = b$ for all $a, b \in \{1, 2, ..., n\}^2$. Moreover, if $a = b$, then $a_i = b_i$ and $\min\{|a_i - b_i|, n - |a_i - b_i|\} = 0$ for all $a, b \in \{1, 2, ..., n\}^2$ and $i \in \{1, 2\}$. Therefore, if $a = b$, then $d'(a, b) = 0$ for all $a, b \in \{1, 2, ..., n\}^2$.

The function d' also has the symmetry property because $|a_i - b_i| = |b_i - a_i|$ and

$$d'(a, b) = \sum_{i=1}^{2} \min\{|a_i - b_i|, n - |a_i - b_i|\} = \sum_{i=1}^{2} \min\{|b_i - a_i|, n - |b_i - a_i|\} = d'(b, a)$$

for all $a, b \in \{1, 2, ..., n\}^2$ and $i \in \{1, 2\}$.

The function d' also has the triangle inequality property, as proved below. Let $c \in \{1, 2, ..., n\}^2$ and $c = (c_1, c_2)$, by the triangle inequality,

$$\begin{aligned} d'(a,b) &\leq d'(a,c) + d'(c,b) \equiv \\ \sum_{i=1}^{2} \min\{|a_i - b_i|, n - |a_i - b_i|\} &\leq \begin{matrix} \sum_{i=1}^{2} \min\{|a_i - c_i|, n - |a_i - c_i|\} + \\ \sum_{i=1}^{2} \min\{|c_i - b_i|, n - |c_i - b_i|\} \end{matrix} \equiv \\ \sum_{i=1}^{2} \min\{|a_i - b_i|, n - |a_i - b_i|\} &\leq \sum_{i=1}^{2} \begin{pmatrix} \min\{|a_i - c_i|, n - |a_i - c_i|\} + \\ \min\{|c_i - b_i|, n - |c_i - b_i|\} \end{pmatrix}. \end{aligned} \tag{2.1}$$

The inequality

$$\min\{|a_i - b_i|, n - |a_i - b_i|\} \leq \min\{|a_i - c_i|, n - |a_i - c_i|\} + \min\{|c_i - b_i|, n - |c_i - b_i|\}$$

must hold for all $a, b, c \in \{1, 2, ..., n\}^2$ and $i \in \{1, 2\}$, by Equation 2.1. For more intuitive notation, let $x_a = a_1$, $x_b = b_1$ and $x_c = c_1$ for $i = 1$ and, without loss of generality, suppose that $x_a \geq x_b \geq x_c$.

- If $x_a - x_b \leq n - (x_a - x_b)$ and $x_a - x_c \leq n - (x_a - x_c)$ hold, then

$$\begin{aligned} \min\{|x_a - x_b|, n - |x_a - x_b|\} &= x_a - x_b \\ &\leq x_a - x_c \\ &= \min\{|x_a - x_c|, n - |x_a - x_c|\} \\ &\leq \min\{|x_a - x_c|, n - |x_a - x_c|\} + \min\{|x_b - x_c|, n - |x_b - x_c|\}. \end{aligned}$$

- If $x_a - x_b \leq n - (x_a - x_b)$, $n - (x_a - x_c) < x_a - x_c$ and $x_b - x_c \leq n - (x_b - x_c)$ hold, then

$$\begin{aligned} \min\{|x_a - x_b|, n - |x_a - x_b|\} &= x_a - x_b \\ &\leq n - (x_a - x_b) \\ &= n - x_a + x_b + x_c - x_c \\ &= n - (x_a - x_c) + x_b - x_c \\ &= \min\{|x_a - x_c|, n - |x_a - x_c|\} + \min\{|x_b - x_c|, n - |x_b - x_c|\}. \end{aligned}$$

- If $x_a - x_b \leq n - (x_a - x_b)$, $n - (x_a - x_c) < x_a - x_c$ and $n - (x_b - x_c) < x_b - x_c$ hold, then

$$\begin{aligned} \min\{|x_a - x_b|, n - |x_a - x_b|\} &= x_a - x_b \\ &\leq x_a - x_b + n - (x_a - x_c) \\ &= n - x_b + x_c \\ &= \min\{|x_b - x_c|, n - |x_b - x_c|\} \\ &\leq \min\{|x_a - x_c|, n - |x_a - x_c|\} + \min\{|x_b - x_c|, n - |x_b - x_c|\}. \end{aligned}$$

- If $n - (x_a - x_b) < x_a - x_b$, then $n - (x_a - x_c) < x_a - x_c$ and $x_b - x_c \leq n - (x_b - x_c)$ hold. Therefore,

$$\begin{aligned} \min\{|x_a - x_b|, n - |x_a - x_b|\} &= n - (x_a - x_b) \\ &= n - x_a + x_b + x_c - x_c \end{aligned}$$

$$
\begin{aligned}
&= n - (x_a - x_c) + x_b - x_c \\
&= \min\{|x_a - x_c|, n - |x_a - x_c|\} + \min\{|x_b - x_c|, n - |x_b - x_c|\}.
\end{aligned}
$$

The proof for $i = 2$ is the same of $i = 1$. As $\min\{|a_i - b_i|, n - |a_i - b_i|\} \leq \min\{|a_i - c_i|, n - |a_i - c_i|\} + \min\{|c_i - b_i|, n - |c_i - b_i|\}$ holds for all $a, b, c \in \{1, 2, ..., n\}^2$ and $i \in \{1, 2\}$, then $d'(a, b) \leq d'(a, c) + d'(c, b)$ also holds and, as a consequence, d' is a metric function on $\{1, 2, ..., n\}^2$. □

The Section 3.1 presents random graph generative models that use grids or tori as the base graph and use the lattice distance or the torus distance to generate random edges. In addition, the same section presents routing algorithms for computer networks that use the lattice distance or the torus distance to select the edge by which to forward a message. Section 4.1 presents a random graph generative model that uses the torus as base graph and the torus distance to generate random edges. Furthermore, the same section presents an analysis that assumes that the torus distance has all properties of Definition 1, as Theorem 10 proves.

2.3 Compact Routing Schemes

Packet routing in computer networks is a classic problem in computer science. In this section, the term "network" is related to computers networks and "graph" is related to the mathematical object that contains a pair of a vertices set and an edges set. Let $G = (V, E)$ be a connected and undirected graph, each vertex $u \in V$ stores a routing table that is used by the routing algorithm to take routing decisions. Each routing table entry has an identifier number of a vertex $v \in V \setminus \{u\}$ and an identifier number of an edge incident to u which leads to a shortest path to v in G. The Definition 11 formalizes the vertex identifier number. Note that the vertex id of a vertex u is denoted in this thesis directly by u rather than by a function. The Definition 12 formalizes the edge identifier number. In the case that G is directed, the notation $\partial(u)$ in Definition 12 corresponds to the **set of out-edges** of u in G and the port number $p(u, v)$ identifies the **directed edge** $(u, w) \in \partial(u)$. Note that p is a total function on $V \setminus \{u\}$ and there may exist two vertices $v, w \in V \setminus \{u\}$ such that $p(u, v) = p(u, w)$.

Definition 11. The **vertex id** u of a vertex $u \in V$ is an identifier number unique among the vertices ids of all vertices in V.

Definition 12. The **port number** $p(u, v)$ in a vertex $u \in V$ to all vertices $v \in V \setminus \{u\}$ is the one of the unique port numbers of all edges in $\partial(u)$ that identifies an edge $\{u, w\} \in \partial(u)$ that leads to a shortest path from u to v in $G = (V, E)$.

There are two extremes in length of paths found by the routing algorithm. The optimal solution consists in (i) routing through shortest paths and the worst solution consists in (ii) routing through a logical unidirectional n-ring. In this sense, routing through shortest paths corresponds to the upper bound on memory consumption per vertex and routing through a ring corresponds to the lower bound. Formalizing the (i) routing all messages along shortest paths, each vertex u forwards messages searching a specific entry in the local routing table with $n - 1$ entries. The notation n corresponds to $|V|$ throughout this section. The vertices ids and port numbers are represented with at most $\lfloor \log_2 n \rfloor + 1$ bits because u may be neighbor of all vertices such that $|\partial(u)| = n - 1$. Therefore, each vertex requires at most $(n - 1) \cdot 2(\lfloor \log_2 n \rfloor + 1)$ bits of memory to store the local routing table, that is $\mathcal{O}(n \log n)$ bits.

Formalizing the (ii) routing all messages along a logical unidirectional n-ring, each vertex stores only its own vertex id, the vertex id of one neighbor and one port number. This solution requires at most $3(\lfloor \log_2 n \rfloor + 1)$ bits in each vertex and the path length is $n - 1$ in the worst case. A **compact** routing scheme, formally presented in Definition 20, requires at most sublinear number of bits in each vertex, that is, $o(n)$ bits (Cowen, 1999; Abraham et al., 2004a; Brady and Cowen, 2006). The Figure 2.6 shows the relation between the two presented extremes in paths length and where compact routing schemes are classified. Note that in routing through a logical unidirectional n-ring, the path found by the routing algorithm may increase in $\mathcal{O}(n)$ **compared to the shortest path**.

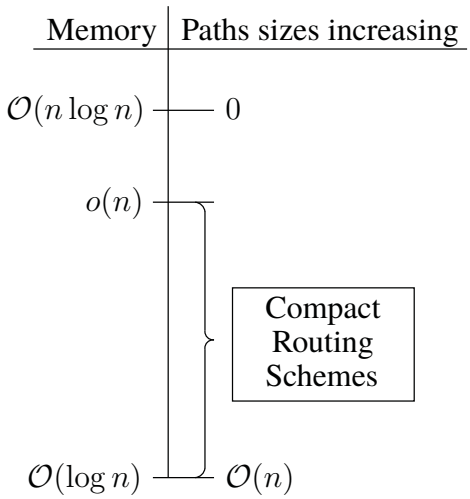
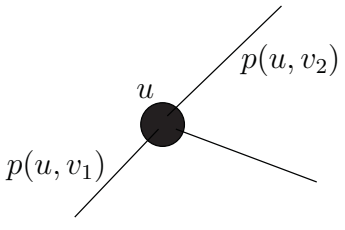


Figure 2.6: Routing tables sizes of compact routing schemes.

2.3.1 Routing Tables

The routing tables assist the routing of messages in each node of the network. Routing tables are data structures that store vertices ids, port numbers and any other network structural information. Let $G = (V, E)$ be a connected and undirected graph, each vertex $u \in V$ stores a local routing table and may store more auxiliary structures if necessary. At each received message, vertex u executes a routing algorithm that uses the local routing table to select by which edge incident to u the message will be forwarded. Figure 2.7 shows the first two entries of the routing table of vertex u. For example, the routing algorithm forwards all messages with target vertex $v_2 \in V$ by the edge incident to u identified by the port number $p(u, v_2)$. Definition 13 formalizes routing tables entries.



Routing table of u		
Vertex	Port	Additional inf.
v_1	$p(u, v_1)$	$\cdots$
v_2	$p(u, v_2)$	$\cdots$
$\vdots$	$\vdots$	$\ddots$

Figure 2.7: Routing table of a vertex $u \in V$.

Definition 13. A **routing table entry** is a triple $(u, p, \mathfrak{i})$, where u is the vertex id of $u \in V$, $p \in \mathbb{N}$ is a port number and $\mathfrak{i}$ is an information.

The routing tables entries are defined by at least two identifier numbers u and p and, if necessary, by any other additional topological information $\mathfrak{i}$ of the network. A routing scheme is **fixed-port** if it allows the network designer to assign the vertices ids and port numbers. In this case, the network designer is free to assign any information, as in Definition 4, to the vertices ids and port numbers beforehand. On the other hand, a routing scheme is **designer-port** if the preprocessing algorithm, explained in Section 2.3.3, outputs the vertices ids and port numbers. The vertices ids u and port numbers p are represented by informations with bounded dimension k, according to Definition 4. It is important to assign wisely the vertices ids and port numbers such that they do not require a large number of bits to represent them. The information $\mathfrak{i}$ also has bounded dimension k and are negligible in some routing schemes, that is, has dimension $k = 0$. An entry has dimension of at least $2(\lfloor \log_2 n \rfloor + 1)$ bits in the best case, where $\lfloor \log_2 n \rfloor + 1$ represent u and $\lfloor \log_2 n \rfloor + 1$ represent p. The definitions of routing table and routing table size follow.

Definition 14. The **routing table** of a vertex $u \in V$ is the set T_u that stores routing table entries.

Definition 15. The **size** in bits of a routing table T is

$$s(T) = \sum_{(u,p,\mathfrak{i}) \in T} (k(u) + k(p) + k(\mathfrak{i})),$$

where $k(x)$ is the dimension of the information x.

2.3.2 Labels and Headers

In **name-dependent** routing schemes, each vertex has a tuple with bounded dimension defined by its vertex id and some more network topology information. These tuples can be seen as "addresses" of the vertices in the network and are named **labels**. In this context, each vertex $u \in V$ also stores its label $l(u)$. The vertex labeling l follows the Definition 5. It is important that the label $l(u)$ of each u requires small number of bits for representation. Definition 16 formalizes small vertex labeling. On the other hand, the vertices of a **name-independent** routing scheme have labels with information unrelated to the network topology. For example, when the label of each vertex is only its vertex id.

Definition 16. The size of a vertex labeling l, denoted by $s(l(u))$, is **small** if

$$s(l(u)) = \sum_{i=1}^{k(l(u))} k(u_i) \text{ is } \mathcal{O}\left(\log^{\lambda} n\right) \text{ for all } u \in V,$$

where $k(x)$ is the dimension of the information/tuple x, $l(u) = \left(u_1, ..., u_{k(l(u))}\right)$ and the constant $\lambda \geq 1$.

A **message** is an information sent by a vertex of the network. A portion of the message is an information used for routing it on the network. Usually, such information is about the target vertex, path and network topology. This routing information in the message is named **header**. It is also important that the header has small number of bits. The Definition 17 formalizes small header.

Definition 17. A message header h is **small** if $k(h)$ is $\mathcal{O}\left(\log^{\lambda} n\right)$, where $k(h)$ is the dimension of the information h and the constant $\lambda \geq 1$.

2.3.3 Preprocessing Algorithms

Preprocessing algorithms generate the vertices ids, port numbers, routing tables, labels and auxiliary data structures for each vertex of the network. Preprocessing algorithms of fixed-port routing schemes take as input a connected graph $G = (V, E)$, a table of vertices ids u and a table of port numbers $p(u, v)$, for all $u, v \in V$. The table of vertices ids follows the Definition 11 and the table of port numbers follows the Definition 12. The algorithm outputs the labels $l(u)$, routing tables T_u and other auxiliary structures, such as dictionaries, of all vertices $u \in V$. Some routing schemes do not require auxiliary structures.

A preprocessing algorithm is **efficient** if its running time is polynomial in the size of the input. Theorem 82, presented in Section 6.6, defines a new compact routing scheme with an efficient preprocessing algorithm. The definition of name-dependent preprocessing algorithm is presented in Definition 18. This definition extends to designer-port routing schemes, where the algorithm does not take as input neither the table of vertices ids nor the table of port numbers, unlike this, it outputs both tables. Moreover, Definition 18 also extends to name-independent routing schemes, where the algorithm does not output any vertex labeling.

Definition 18. A **name-dependent preprocessing algorithm** takes as input a connected graph $G = (V, E)$, a table of vertices ids u and a table of port numbers $p(u, v)$ for all $u, v \in V$ and outputs a vertex labeling l in G, the set $\mathcal{T}$ of all routing tables T_u and all other auxiliary structures.

2.3.4 Routing Algorithms

The same routing algorithm executes in all vertices of the network and forwards messages that reach the vertices. The routing algorithm of a name-dependent scheme takes as input the header h of the received message, the label $l(u)$ of the local vertex $u \in V$, the routing table T_u of u and all other auxiliary structures. It outputs the port number of the edge that the message will be forwarded, if u is not the target. If u is the target, then it outputs the entire message and does not forward the message to any neighbor. A routing algorithm is **efficient** if its running time is constant. It is important that the routing algorithm be efficient such that there is not a high latency in the message routing. The definition of name-dependent routing algorithm follows. Definition 19 extends to name-independent routing schemes, where the algorithm does not take as input the label $l(u)$ of u.

Definition 19. A **name-dependent routing algorithm** takes as input a vertex id u, a label $l(u)$, a routing table T_u and other auxiliary structures of a vertex $u \in V$ and outputs the port number $p(u, v)$ that identifies the edge in $\partial(u)$ by which the message will be forwarded.

2.3.5 Efficiency Measurements

Compact routing schemes must satisfy some constraints. The preprocessing and routing algorithms must be efficient, according to the Sections 2.3.3 and 2.3.4. The message headers must be small, according to Definition 17. The vertex labeling function of the name-dependent schemes must be small, according to Definition 16. The sum of the sizes of the label, routing table and all other auxiliary structures stored in each vertex must be sublinear in the number of vertices. The definition of compact routing scheme follows. The size $s(l(u))$ of the label $l(u)$ of a vertex $u \in V$ is in Definition 16 and the size $s(T_u)$ of the routing table T_u of u is in Definition 15.

Definition 20. A routing scheme is **compact** if $s(l(u)) + s(T_u) + s(S)$ is $o(n)$ for all $u \in V$, where $s(l(u))$ is the size of the label $l(u)$ of u, $s(T_u)$ is the size of the routing table T_u of u and $s(S)$ is the dimension of the information S that encodes all auxiliary structures of u.

The length of the paths by which the routing algorithm routes the messages also measures the efficiency of compact routing schemes. The two metrics related to the paths length are the stretch factor, formalized in Definition 21, and the expected paths length, formalized in Definition 22.

The stretch factor is an upper bound multiplicative factor of the increasing regarding the length of a minimum weighted path. For example, when a compact routing scheme has stretch factor 3, the length of any path found by the routing algorithm is at most 3 times the length of a minimum weighted path. A routing scheme with stretch factor 1 always routes by a minimum weighted path. Let $G = (V, E)$ be a connected and undirected graph with weights in the edges that defines a metric space. Let $d(u, v)$ be the length of a minimum weighted path from all vertices $u \in V$ to all vertices $v \in V$ in G. Let $d'(u, v)$ be the length of the weighted path from u to v that a message passed on, through the routing algorithm. The definition of stretch factor follows. Definition 21 also extends to unweighted graphs, where d and d' are the number of edges of the paths.

Definition 21. The **stretch factor** of a compact routing scheme is

$$\max_{u,v \in V} \left(\frac{d'(u, v)}{d(u, v)} \right), \text{ where } u \neq v.$$

The expected paths length is the average of the lengths of all paths that the routing algorithm may find. The routing algorithm may find paths greater or less than the expected paths length. The length of a path is modeled by a random variable X and the expected paths length is the expected value $\mathrm{E}[X]$ of X. It is also important that the expected paths length of a compact routing scheme is small. The definitions of expected paths length and small paths follow.

Definition 22. The **expected paths length** of a compact routing scheme is the expected value $\mathrm{E}[X]$ of the random variable X of the length of the path that the routing algorithm finds.

Definition 23. A path with length defined by the random variable X is **small** if $\mathrm{E}[X]$ is $\mathcal{O}\left(\log^{\lambda} n\right)$ for a constant $\lambda \geq 1$.

2.4 Random Small World Graphs

Milgram (1967) presents one of the first work in social networks. He spread letters across the United States of America containing little information about a destination in the same country. The people who received a letter were asked to forward it to an acquaintance closest to the destination. The goal of this experiment is to measure the average of forwardings necessary to a letter to reach its destination and, thereby, to estimate how far apart people are from each other in social networks. The main conclusions are: (i) there are many short paths between different pairs of people in social networks, with 6 hops in average; (ii) everyone performs a local search without knowing the entire network and (iii) people find short paths collectively through sequences of decentralized local searches. The Milgram experiment shows that social networks have many short paths that interconnect pairs of people. This topological property in social networks is named **small world phenomenon**.

Small world graphs have small paths that interconnect most pairs of vertices. There are mathematical models that generate random small world graphs. Section 3.1 presents some of them and Sections 4.1 and 5.2 propose two new small world models. Mathematical models allow the theoretical analysis of algorithms executing on graphs generated by them. The Chapter 6 proposes and analyzes an algorithm that executes on graphs generated by the small world model presented in Section 4.1. They also allow to compute the upper bound on the expected value of the length of the paths. For example, the model presented by Kleinberg (2000a) and described in Section 3.1, generates graphs $G = (V, E)$ with paths of $\mathcal{O}(\log |V|)$ expected length. The Definition 24 formalizes small world graphs.

Definition 24. A graph $G = (V, E)$ is a **small world graph** if the random variable X of the length of a path in G has expected value $\mathrm{E}[X] = \mathcal{O}(\log |V|)$.

Watts and Strogatz (1998) claim that many biological, technological and social networks have topology between completely regular and completely random. They present a generative model that allows the tuning of this trade-off through a probability value. The model has three parameters: the number of vertices $n \geq 3$, the even number of local contacts $2 \leq p \leq n - 1$ and the probability $0 \leq \beta \leq 1$. It starts generating a n-ring lattice which each vertex has undirected edges to its p closest neighbors in the lattice without generating parallel edges. After generating this regular structure, it rewires each edge once by the following way. The model chooses an initial vertex and the edge to its closest neighbor in clockwise. This neighbor is changed with probability β to another vertex chosen uniformly at random, if this does not generate duplicate edges neither loop. This procedure is repeated for all vertices in the clockwise sequence until reaching the initial vertex. Then, the model performs one more lap, rewiring the edges that connect to the second closest vertices and so on, until complete $p/2$ laps.

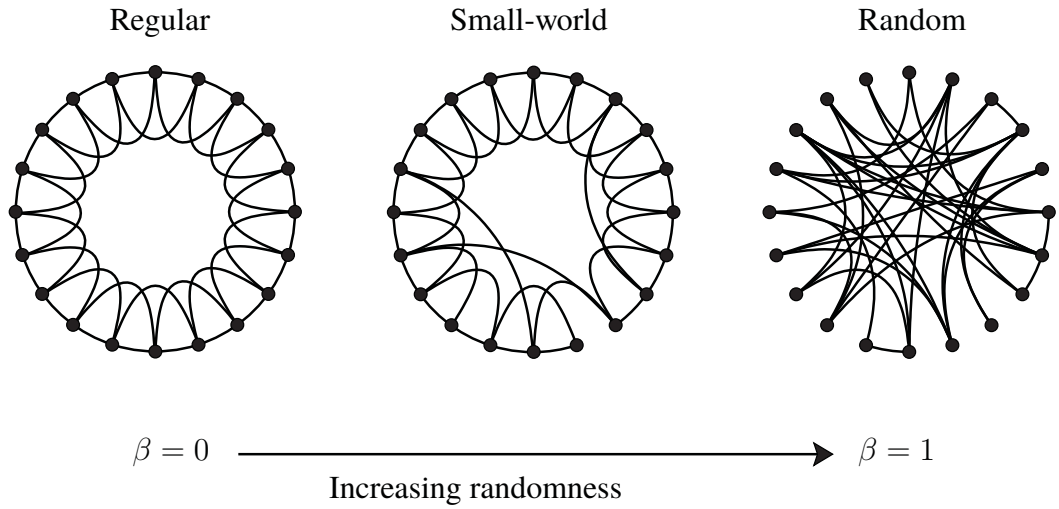


Figure 2.8: Graphs with $n = 20$, $p = 4$ and distinct values of β (Watts and Strogatz, 1998).

This model generates from completely regular graphs, for $\beta = 0$, to completely random, for $\beta = 1$, as Figure 2.8 shows. It creates a regular base graph to implement clustering and changes the edges randomly to implement small paths. Watts and Strogatz experimentally show that the graphs generated by their model have both structural properties for some values of β less than 0.1. It is considered a small world model because it can generate graphs with small average of the shortest paths between all pairs of vertices. Many models use the idea of initially generating a regular base graph following of changing or adding edges at random, implementing both clustering and small paths.

2.5 Small and High Probabilities

When a convergent function in a real number n bounds the probability of an event occurring, then it is possible to compute the probability value when n goes to infinity. An event occurs with small probability when it is "close" to 0 for "large" values of n. On the other hand, an event occurs with a high probability when it is "close" to 1 for "large" values of n. The definitions of small and high probabilities follow. Chapters 4, 5 and 6 use both definitions to describe probabilities of events occurrence.

Definition 25. Let $\mathcal{E}$ be an event such that $\Pr(\mathcal{E}) \leq f(n)$. The event $\mathcal{E}$ occurs **with small probability** if $\lim_{n\to\infty} f(n) = 0$.

Definition 26. Let $\mathcal{E}$ be an event such that $\Pr(\mathcal{E}) \geq f(n)$. The event $\mathcal{E}$ occurs **with high probability** if $\lim_{n\to\infty} f(n) = 1$.

3 Literature Review

This chapter presents the literature review about routing algorithms in distinct small world models and compact routing schemes that routes messages through paths with bounded stretch. The next two sections begin presenting with more details a well known solution in the literature and follow with an abstract presenting the evolution of the state of the art. Both sections also present a table in the end (Tables 3.1 and 3.2) that summarizes the main contributions in the same sequence as they are presented. The last section presents more works with complementary information about the previous sections.

The Section 3.1 presents works that define small world models and greedy routing algorithms that execute in graphs generated by such models. Generating a base graph to model clustering and generating random edges to model small paths are the main strategies used in most of the works. The presented models generate the random edges with a specific probability distribution proposed by Kleinberg (2000a) that allows the greedy routing to find small paths. In general, the works present small variations in the model and in the routing algorithm, achieving better results in the lengths of the paths that greedy routing finds without increasing the routing tables sizes.

The Section 3.2 presents works in routing scheme that bound the maximum length of the paths found by the routing algorithm in function of the minimum path length. The authors of the cited works analyze their routing schemes, proving small sizes of routing tables, vertices labels and message header, while reducing the stretch factor. Some adopted strategies are optimizing procedures of already known routing schemes, combining distinct routing schemes and designing routing schemes specialized in specific classes of graphs.

All works presented throughout Section 3.1 deal with **directed and unweighted** graphs. On the other hand, the graphs are **undirected and have weights in the edges** throughout Section 3.2. The weights are positive and define a metric space, as presented in Section 2.1. Each edge has weight 1 when there is no mention about weights, also defining a metric space as Theorem 2 proves. Also, d is the standard notation for distance functions in both sections. Each small world model defined in Section 3.1 is completely dependent from the distance function. Hence, the distance function is redefined locally for each presented small world model. In Section 3.1, the distance function is defined as the length of the minimum weighted path. Finally, the term “network” is related to social and computers networks and “graph” is related to a pair defined by a set of vertices and a set of edges.

3.1 Routing in Small World Models

The work of Kleinberg (2000a) represents a breakthrough on the research of small world networks and routing. He solves a computational problem that is open since the Milgram experiments in 1967 (Milgram, 1967). Milgram shows that people in a social network are able to find small paths working together in a decentralized way. Despite that, Watts and Strogatz (1998) present a

model that generates graphs with a large number of small paths, no routing algorithm capable to find these paths with only little local information was known until that moment. Kleinberg defines a probability distribution for the random edges generation and present a fast greedy routing algorithm that finds small paths in graphs generated with this distribution. In other words, he presents an algorithmic model of the Milgram experiments, where computers in a small world network exchange messages through small paths without knowing the entire network neither executing complex routing algorithms.

Kleinberg uses the ideas of Bollobás and Chung (1988) for the small world graphs generation. Bollobás and Chung prove that a n-cycle including a random matching has diameter $\Theta(\log n)$ with high probability. Kleinberg (2000a) defines a model that adds more edges to the base graph, instead of modifying the already generated base graph edges, as in the Watts and Strogatz' model. The model generates a $n \times n$ lattice of vertices such that $V = \{1, 2, ..., n\} \times \{1, 2, ..., n\}$, where $n \geq 1$ is the graph size parameter. He defines the **lattice distance** $d((i, j), (k, l)) = |k - i| + |l - j|$ as the number of "lattice steps" between each pair of vertices $(i, j), (k, l) \in V$. For each vertex, the model generates directed edges to all vertices within lattice distance $p \geq 1$. After creating this base graph, the model generates $q \geq 0$ more edges for each vertex using independent random trials. A vertex $u \in V$ has one of the q directed edges to $v \in V \setminus \{u\}$ with probability proportional to $d^{-r}(u, v)$, where $r \geq 0$. Moreover, the probability of each $v \in V \setminus \{u\}$ is multiplied by the **normalizing factor** of u

$$\left(\sum_{w \in V \setminus \{u\}} d^{-r}(u, w) \right)^{-1}$$

in order to obtain a probability distribution. Kleinberg names this probability distribution as the **inverse r^{th}-power distribution**. The edges generated by this distribution are referred as **long-range edges**. The Figure 3.1 (A) presents the base graph generated with $n = 6$ and $p = 1$ and (B) presents a vertex $u \in V$, its four edges to the vertices within lattice distance $p = 1$ and its $q = 2$ long-range edges to $v \in V \setminus \{u\}$ and $w \in V \setminus \{u\}$.

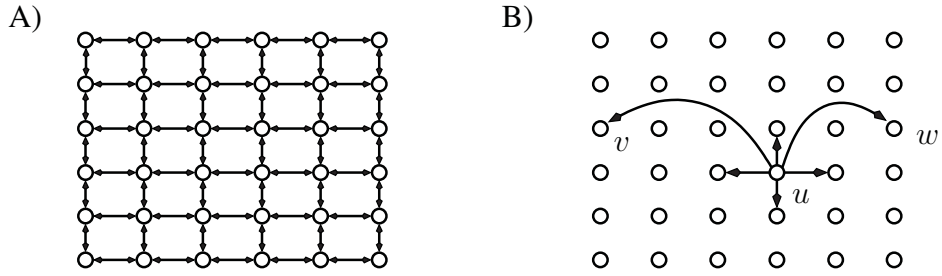


Figure 3.1: A base graph (A) and the neighborhood of a vertex (B) (Kleinberg, 2000a).

Kleinberg presents a greedy routing algorithm that uses only information about the neighborhood in messages forwarding. Besides that, the complexity of the algorithm when it executes in a vertex is independent on the size n of the graph. It forwards a message in a vertex $u \in V$ to a neighbor $v \in \mathcal{N}^+(u)$ with minimum lattice distance $d(v, t)$ from the target vertex $t \in V$, where $\mathcal{N}^+(u)$ is the set of out-neighbors of u. It extracts the target vertex position in the lattice from the message header, computes the distances in the lattice between each neighbor and the target, and selects the neighbor to forward the message. Each vertex stores only the positions in the lattice of its neighbors, which are pairs of values of $\{1, 2, ..., n\}$. This algorithm is a good alternative solution compared to a preprocessed breadth-first search, which requires storing the entire graph in each vertex to perform a routing decision. The effectiveness of this alternative

depends only on the values of p and q, and on the length of the paths that the greedy routing algorithm finds with respect to the graph diameter.

Kleinberg also proves that for $p = q = 1$ and $r = 2$, the greedy routing algorithm delivers messages through paths of $\mathcal{O}\left(\log^2 n\right)$ expected length. This is a good result because p and q are low constants, requiring low constant storage in each vertex. Besides that, the length of the path that the algorithm finds is polynomial on the expected diameter. Martel and Nguyen (2004) prove that the expected diameter of the graph that the Kleinberg's model generates is $\Theta(\log n)$. Kleinberg also shows that $r = 2$ is the only value where the paths lengths are not exponential in $\log n$ and he claims that, for models of dimensions $k \geq 1$, similar results are obtained for $r = k$. Martel and Nguyen (2004) also prove that the expected diameter is also $\Theta(\log n)$ for $r = k$. These results also present another important property of computational small world networks which is that minimizing the diameter of the network does not imply in minimizing the network transmission rate. Despite the network having small diameter, the decentralized routing may not be capable to find small paths to the target. Kleinberg also describes these results in another paper (Kleinberg, 2000b).

Many works present variations in the Kleinberg's model or in its greedy routing algorithm and achieve other results. Barrière et al. (2001) analyze the performance of the greedy routing algorithm executing on a unidirectional n-ring with one directed long-range edge per vertex generated with the inverse first-power distribution. They define the distance from a vertex $u \in V$ to a vertex $v \in V$ as the number of directed edges of the shortest path on the ring with start vertex u and final vertex v. They prove a tight bound of $\Theta\left(\log^2 n\right)$ for the expected length of the paths that the greedy routing finds. Then, Barrière et al. show that the Kleinberg's model can be simplified to a base unidirectional ring.

Manku et al. (2003) present the Symphony protocol for maintaining distributed hash tables in peer-to-peer networks. Distributed Hashing Protocols (DHP) map a hash table into hosts of a network, and the number of hops in searches is an important performance metric to reduce. Symphony is inspired on the Kleinberg's construction, managing a logical small world network with constant number of edges per vertex, and paths with small length in the greedy routing. The network building is similar to the model of Barrière et al., but in Symphony it is possible to choose the number $q = \mathcal{O}(1)$ of long-range edges. Manku et al. prove that the expected length of the paths that the greedy routing finds in both Symphony, with directed and undirected edges, is $\mathcal{O}\left(\frac{\log^2 n}{q}\right)$. In other work, Manku et al. (2004) present a greedy routing algorithm that considers the vertex neighbors and the neighbors of neighbors in a routing decision. They prove that this improved version of the routing algorithm finds paths with $\mathcal{O}\left(\frac{\log^2 n}{q \cdot \log q}\right)$ expected length for $1 \leq q \leq \log n$ long-range edges per vertex using the Symphony with directed edges. Aspnes et al. (2002) present a protocol similar to Symphony.

Lebhar and Schabanel (2004) create a model that uses a base torus generated over the k-dimensional lattice $\{-n, ..., 0, ..., n\}^k$ with undirected edges. The model generates q directed long-range edges per vertex with the inverse k^{th}-power distribution and the distance function defined by the length of the shortest path on the torus. Lebhar and Schabanel present a decentralized routing algorithm that performs a sequence of **breadth-first search forwardings** and uses the Kleinberg's greedy routing in the last portion of the path. Each breadth-first search forwarding performs a specific breath-first search with bounded depth rooted in the local vertex $u \in V$. Then, it finds the vertex $v \in V$ closest to the target $t \in V$ in the torus and sends the message through the shortest path to v found. When the message reaches vertex v, it executes a new breadth-first search forwarding to the next vertex closest to t, and so on. The decentralized routing algorithm iteratively executes these forwardings while the distance in the torus from the local vertex u to t is greater than $q \cdot \log^2 n$. The algorithm executes Kleinberg's greedy

routing when u is within distance $q \cdot \log^2 n$ from t. Lebhar and Schabanel prove that their routing algorithm visits $\mathcal{O}\left(\frac{\log^2 n}{\log^2(1+q)}\right)$ expected number of vertices in the executions of all breadth-first search forwardings, and routes through paths of $\mathcal{O}\left(\frac{\log n \cdot \log^2 \log n}{\log^2(1+q)}\right)$ expected length.

Martel and Nguyen (2004) present a routing algorithm and analyze its performance on the Kleinberg's model over a lattice with $k \geq 1$ dimensions together with k-dimensional lattice distance. The algorithm considers that each vertex stores the position in the lattice of the endpoints of the long-range edges of its $\log n$ closest vertices in the lattice. It performs iteratively a sequence of forwardings between intermediate destinations $u \in V$ while the message does not reach the target. The start vertex $s \in V$ routes the message towards the vertex $u \in W_s$ closest to the target $t \in V$ in the lattice, where W_w is the set of the $\log n$ vertices closest to $w \in V$ and the heads of their respective long-range edges. While the message is in a vertex in the sub-path from s to u, the intermediate vertex sends the message only through edges of the base graph. When the message reaches u, it routes the message to another intermediate destination $v \in W_u$ and so on, until the message reaches t. Martel and Nguyen show that, for $p = q = 1$, their algorithm finds paths with $\mathcal{O}\left(\log^{1+1/k} n\right)$ expected length, and they also claim that this result extends to a base torus. This algorithm is a good alternative for routing when the dimension k is large.

Fraigniaud et al. (2006) present a greedy routing algorithm that also considers near long-range edges, but it is completely **oblivious**. An oblivious algorithm does not change the information encoded in the message header in all forwardings. The routing algorithm presented by Martel and Nguyen (2004), for example, encodes a path in the message header so that each intermediate vertex can forward the message to the next vertex of the path. This non-oblivious algorithm changes the path encoded in the header as the message pass through the vertices of the graph, different, for example, from the oblivious Kleinberg's greedy routing algorithm. The main motivation of Fraigniaud et al. in designing an oblivious algorithm is that the Milgram's experiment was conducted by an oblivious way. Besides, they also claim that obliviousness is a property that may ensures better fault-tolerance in computer networks.

The greedy routing algorithm of Fraigniaud et al. computes an intermediate destination $v \in V$ in each local vertex $u \in V$ that the message reaches. Let A_u be the set of the long-range edges of the $\log n$ closest neighbors of u in the lattice. The algorithm selects all edges $(a, b) \in A_u$ such that the lattice distance from $b \in V$ to the target $t \in V$ is minimum. If there are two or more edges, then it selects whose that the lattice distance from u to $a \in V$ is minimum. If, even so, there are two or more, then it selects one arbitrarily. Once (a, b) is selected from A_u, if $a = u$ or the lattice distance from u to a is at least the lattice distance from u to t, then the algorithm sets the intermediate destination $v = t$; or sets $v = a$ otherwise. The algorithm forwards the message from u to its out-neighbor closest to v in the lattice. Note that each vertex u that receives the message computes its own intermediate destination v and forwards the message without changing the message header. Fraigniaud et al. prove the same results of Martel and Nguyen (2004) through this oblivious algorithm executing on the Kleinberg's model over the k-dimensional lattice for $p = q = 1$ and $r = k$. Moreover, they prove that defining A_u with the distance $\log n$ is the best fitting to minimize the expected paths lengths. The same authors had already published these results some years ago in a technical report (Fraigniaud et al., 2003).

Zeng et al. (2005) present a small world model over the unidirectional n-ring where each vertex $u \in V$ has one directed long-range edge generated with the inverse first-power distribution. They define the distance function as the length of the shortest directed path on the ring between the vertices. Moreover, the model generates two more directed **augmented local edges** from each vertex u to vertices within distance $\log^2 n$ from u chosen uniformly at random. They also define the **awareness** $A_u(i)$ of u as the set of all vertices within distance i from u that

are reachable only through augmented local edges. Zeng et al. present both non-oblivious and oblivious algorithms that route messages with expected $\mathcal{O}(\log n \cdot \log\log n)$ forwardings.

The non-oblivious algorithm computes iteratively a sequence of continuous paths, forwarding the message through them, while the distance from the current message holder $u \in V$ to the target $t \in V$ is at least $\log^2 n \cdot \log\log n$. In this case, in the beginning, u selects the intermediary target $v \in A_u(\log\log n)$ such that the distance from $w \in V$ to t is minimum, where $(v, w) \in E$ is the long-range edge of v and ties are broken arbitrarily. Then, u computes the shortest path to v composed only by augmented local edges, encodes the path into the message header and forwards the message to the next vertex of the path. The subsequent vertices only decode the path from the header and forwards the message to the next vertex in the path. When the message reaches the intermediary target v, v forwards the message through its long-range edge to w, w computes a new shortest path to another intermediary target and so on. In the cases where the distance from the current message holder u to the target t is less than $\log^2 n \cdot \log\log n$, the algorithm executes the Kleinberg's greedy routing algorithm.

The oblivious algorithm of Zeng et al. (2005) also has two distinct routing procedures, where one of them is chosen based on the distance from the message holder $u \in V$ to the target $t \in V$. When the distance is at least $\lambda \cdot \log^2 n \cdot \log\log n$, u tries to find a vertex $v \in A_u(\log\log n)$ such that the distance from $w \in V$ to t is at most the half of the distance from u to t. In this context, λ is a sufficiently large constant and $(v, w) \in E$ is the long-range edge of v. If there is no such vertex v, then u forwards the message greedily as in Kleinberg's algorithm. If there are two or more possible choices for v, then the algorithm selects the vertex closest to u based on the number of augmented local edges and if, even so, there are more than one choice, then the algorithm selects one arbitrarily. If the choice is $v = u$, then u forwards the message through its long-range edge, otherwise, u computes a shortest path to v composed only by augmented local edges and forwards the message to the next vertex in the shortest path. When the distance from u to the target t is less than $\lambda \cdot \log^2 n \cdot \log\log n$, u forwards the message greedily as in Kleinberg's algorithm. Note that this algorithm does not change the message header.

One year later, Zeng and Hsu (2006) present another model that builds the base torus with undirected edges over the k-dimensional lattice with size n, for $k \geq 2$. This model also generates one directed long-range edge and two directed augmented local edges per vertex $v \in V$. However, it generates the long-range edge with the inverse k^{th}-power distribution, and each augmented local edge with tail v has head in a vertex within distance $\log^{2/k} n$ from v chosen uniformly at random. Zeng and Hsu define the lattice distance d as the standard distance function in their work. They adapt their oblivious algorithm presented one year ago (Zeng et al., 2005) to that model. The oblivious algorithm of 2006 differ from the 2005 in only three aspects: (i) the distance threshold from the message holder $u \in V$ to the target $t \in V$ that defines routing through continuous paths or routing with Kleinberg's algorithm is $\log^{2/k+1} n$; (ii) the vertices awareness are $A_v(\lambda \cdot \log\log n)$ for all $v \in V$ and $1 < \lambda < 2$; (iii) u tries to find a vertex $v \in A_u(\lambda \cdot \log\log n)$ such that $d(w, t) \leq d(u, t)/\log^{\lambda'} n$, where $(v, w) \in E$ is the long-range edge of v and λ' is a constant. Zeng and Hsu prove that the modified oblivious algorithm routes messages through paths with $\mathcal{O}(\log n)$ expected length. They also claim that this algorithm is optimal because any graph whose vertices with degree $\mathcal{O}(1)$ has $\Omega(\log n)$ expected paths length.

The number of bits required in each vertex to execute a routing algorithm is also an important metric for analysis. In Kleinberg's greedy routing algorithm, for example, each vertex stores the position in the lattice of its at most four neighbors adjacent in the lattice and at most one neighbor reachable through the long-range edge, for $p = q = 1$. Each position in the lattice is defined by a pair of numbers in $\{1, 2, ..., n\}$ and each number in $\{1, 2, ..., n\}$ is represented with $\lfloor \log_2 n \rfloor + 1$ bits. The port number of each out-edge of a vertex, according to Definition 12,

is represented by at most $\lceil \log_2 5 \rceil = 3$ bits. So, the routing tables entries are an information, according to Definition 4, of dimension at most $2(\lfloor \log_2 n \rfloor + 1) + 3$. Therefore, Kleinberg's greedy routing algorithm requires only $\mathcal{O}(\log n)$ bits in each vertex for $p = q = 1$. Table 3.1 summarizes the trade-off between memory consumption per vertex in bits and expected paths lengths of each cited routing algorithm in this section.

Table 3.1: Memory required $\times$ paths length trade-offs (Zeng et al., 2005; Zeng and Hsu, 2006).

Authors	**Obliviousness**	**Number of bits**	**Paths length**
Kleinberg (2000a)	Oblivious	$\mathcal{O}(\log n)$	$\mathcal{O}\left(\log^2 n\right)$
Barrière et al. (2001)	Oblivious	$\mathcal{O}(\log n)$	$\mathcal{O}\left(\log^2 n\right)$
Aspnes et al. (2002)	Oblivious	$\mathcal{O}(q \log n)$	$\mathcal{O}\left(\log^2 n/q\right)$
Manku et al. (2003)	Oblivious	$\mathcal{O}(q \log n)$	$\mathcal{O}\left(\log^2 n/q\right)$
Manku et al. (2004)	Non-oblivious	$\mathcal{O}\left(q^2 \log n\right)$	$\mathcal{O}\left(\log^2 n/(q \log q)\right)$
Lebhar and Schabanel (2004)	Non-oblivious	$\mathcal{O}\left(\log^2 n/\log(1+q)\right)$	$\mathcal{O}\left(\log n(\log\log n/\log(1+q))^2\right)$
Martel and Nguyen (2004)	Non-oblivious	$\mathcal{O}\left(\log^2 n\right)$	$\mathcal{O}\left(\log^{1+1/k} n\right)$
Fraigniaud et al. (2006)	Oblivious	$\mathcal{O}\left(\log^2 n\right)$	$\mathcal{O}\left(\log^{1+1/k} n\right)$
Zeng et al. (2005)	Non-oblivious	$\mathcal{O}\left(\log^2 n\right)$	$\mathcal{O}(\log n \log\log n)$
Zeng et al. (2005)	Oblivious	$\mathcal{O}\left(\log^2 n\right)$	$\mathcal{O}(\log n \log\log n)$
Zeng and Hsu (2006)	Oblivious	$\mathcal{O}\left(\log^{\lambda+1} n\right)$, $1<\lambda<2$	$\mathcal{O}(\log n)$

3.2 Routing with Compact Structures and Bounded Stretch

Thorup and Zwick (2001) present a name-dependent compact routing scheme with stretch factor 3 based on the routing scheme presented by Cowen (1999). Thorup and Zwick reduce the number of bits for the routing table in each vertex from $\mathcal{O}\left(n^{2/3} \cdot \log^{4/3} n\right)$ to $\mathcal{O}\left((n \cdot \log n)^{1/2}\right)$, or simply $\widetilde{\mathcal{O}}\left(n^{1/2}\right)$ removing polylogarithmic factors, where a polylogarithmic function is a polynomial in $\log n$. In this section, n denotes the number of vertices of the graph, that is $n = |V|$. The preprocessing algorithm of this routing scheme generates the label and routing table of each vertex based on a set of vertices called **centers**. The strategic selection of the centers implies in reduction of routing tables sizes. The centers selection algorithm and its respective analysis are in the work of Thorup and Zwick (2001). In what follows, it is given a brief explanation of the compact routing scheme with stretch factor 3 presented by Thorup and Zwick, referred throughout this and next section as **TZ universal scheme**.

Let $G = (V, E)$ be an undirected and weighted graph that defines a metric space in G, according to Section 2.1.1. Such weighted graphs are referred in this section as **metric spaces**. Let $A \subseteq V$ be the set of the already selected centers; $c(u)$ be the vertex in A with minimum weighted path to $u \in V$ in G, where ties are broken arbitrarily, and $p(u, v)$ be the **port number** that identifies the first edge of a minimum weighted path from u to v in G. Port numbers and **vertices ids** are given in advance and are represented by a unique identity number with at most $\lfloor \log_2 n \rfloor + 1$ bits, according to Definitions 12 and 11 respectively. The preprocessing algorithm starts executing the centers selection algorithm that chooses at most $2(n \cdot \log n)^{1/2}$ vertices as centers. After that, it **labels** each vertex $u \in V$ with the triple $(u, c(u), p(c(u), u))$, that is, the id of u, the id of the center $c(u)$ closest to u and the port number that leads to a minimum weighted

path from $c(u)$ to u. The labels are represented by an information, according to Definition 4, of dimension at most $3(\lfloor \log_2 n \rfloor + 1)$. The preprocessing algorithm executes the Dijkstra's algorithm once in the labeling of each vertex.

The preprocessing algorithm generates the routing table of each vertex u after the labeling procedure. The **routing table entries** of u are pairs $(v, p(u, v))$, that is, the id of a stored vertex $v \in V$ and the port number in u that leads to a minimum weighted path to v. The entries are represented by an information, according to Definition 4, of dimension at most $2(\lfloor \log_2 n \rfloor + 1)$. The routing tables of all u store the ids of all centers $a \in A$ and port numbers $p(u, a)$. Let $d(i, j)$ be the length of a minimum weighted path between the vertices $i, j \in V$ in G and

$$C(u) = \{w \in V | d(u, w) < d(c(w), w)\} \tag{3.1}$$

be the **clusters** of all u. The cluster of u contains all vertices in V that are closer to u than the closest center of u. The routing tables of each u also store the ids of all vertices $b \in C(u)$ and port numbers $p(u, b)$. In other words, the **routing table** T_u of u stores all centers and all vertices in its cluster, that is,

$$T_u = \{(a, p(u, a)) \text{ for all } a \in A\} \cup \{(b, p(u, b)) \text{ for all } b \in C(u)\}.$$

Therefore, the number of bits of the routing tables of each u is at most $(|A|+|C(u)|)\cdot 2(\lfloor \log_2 n \rfloor+1)$ because $|T_u| = |A| + |C(u)|$. Thorup and Zwick (2001) prove that $|C(u)| \leq 4(n \cdot \log n)^{1/2}$ if A is an output of their centers selection algorithm. Then, the sizes of all routing tables are $\mathcal{O}\left(n^{1/2} \cdot \log^{3/2} n\right)$ bits. They also present an optimized version of this routing scheme for environments that allows the preprocessing algorithm renaming the vertices ids and port numbers. The optimized TZ universal scheme reduces the labels sizes to $(1 + o(1)) \cdot \log n$ bits and routing tables sizes to $\mathcal{O}\left((n \cdot \log n)^{1/2}\right)$ bits.

The **header** of a message with source vertex $s \in V$ and target vertex $t \in V$ is the label of t, that is $(t, c(t), p(c(t), t))$. Suppose that a message reaches vertex $w \in V$, where $w = s$ if w is the source. The routing algorithm extracts t from the header and verifies whether $w = t$ and if this is the case, then the message reaches the target. Otherwise, it tries to find the entry $(t, p(w, t))$ in the local routing table T_w. If $(t, p(w, t)) \in T_w$, then it forwards the message in w through the edge with port number $p(w, t)$. If $(t, p(w, t)) \notin T_w$, then it verifies whether w is the center closest to the target t, that is $w = c(t)$, where $c(t)$ is also extracted from the message header. If $w = c(t)$, then it forwards the message in w through the edge with port number $p(w, t)$, also extracted from the message header. If $w \neq c(t)$, then it finds the entry $(c(t), p(w, c(t)))$ in the local routing table T_w and forwards the message in w through the edge with port number $p(w, c(t))$. Note that the entry $(c(t), p(w, c(t))) \in T_w$ because all routing tables of all vertices u store all vertices in A.

The routing algorithm is based on routing through a minimum weighted path to the center $c(t)$ closest to the target t and, after that, routing through a minimum weighted path to t. If target t is not w and is in the routing table of w, then $t \in A$ or $t \in C(w)$, trivially and the algorithm routes the message through a minimum weighted path to t. Otherwise, the algorithm routes the message through a minimum weighted path to $c(t) \in A$. If w is $c(t)$, then the algorithm forwards the message to vertex w' in a minimum weighted path to t. There exists an entry $(t, p(w'', t))$ in the routing table $T_{w''}$ of each vertex w'' in a minimum weighted path from w' to t because $t \in C(w'')$, given the Equation 3.1. Figure 3.2 shows a routing performed through the center $c(t)$ closest to t. More details about the correctness of this routing algorithm can be found in the work of Cowen (1999). This algorithm executes the searches in routing tables in $\mathcal{O}(1)$

worst case time if the data structure is the 2-level hash table presented by Fredman et al. (1984). Therefore, this routing algorithm executes in constant time and it is oblivious.

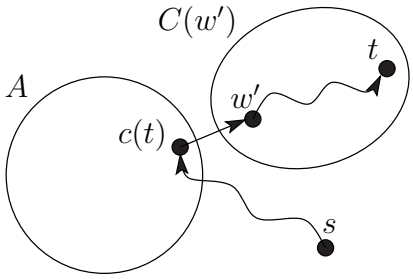


Figure 3.2: An execution of Thorup and Zwick's routing algorithm.

The worst case of path length occurs when the routing algorithm routes the message through a minimum weighted path from s to $c(t)$ and, after that, through a minimum weighted path from $c(t)$ to t. Equation 3.2 begins the algebra of the stretch factor. As the algorithm first routes the message to $c(t)$, so $t \notin C(s)$ and, by Equation 3.1, $d(c(t), t) \leq d(s, t)$. Equation 3.3 follows by the triangle inequality property of d.

$$\begin{aligned} d(s, c(t)) + d(c(t), t) &\leq d(s, c(t)) + d(s, t) && (3.2)\\ &\leq d(c(t), t) + d(s, t) + d(s, t) && (3.3)\\ &\leq d(s, t) + 2 \cdot d(s, t)\\ &= 3 \cdot d(s, t). \end{aligned}$$

Erdős (1963) presents the **girth conjecture** which for all $k \geq 1$, there exists graphs with n vertices and $\Omega\left(n^{1+1/k}\right)$ edges whose girths have size of at least $2k + 2$. Thorup and Zwick claim that if this family of graphs does exist, then every routing scheme with stretch factor less than $2k + 1$ would have at least $\Omega\left(n^{1+1/k}\right)$ bits of total information. In this sense, for $k = 1$, a routing scheme with stretch factor less than 3 would have at least $\Omega(n)$ bits in at least one of the n routing tables, which is not considered compact by Definition 20. In fact, girth conjecture holds for $k = 1$ and, besides that, Gavoille and Gengler (2001) prove that there exists graphs with n vertices in which every routing algorithm of stretch factor less than 3 requires at least $\Omega\left(n^2\right)$ bits of routing information. Therefore, Thorup and Zwick achieve the minimum constant stretch factor for a compact routing scheme.

Thorup and Zwick (2001) also present a compact routing scheme that generalizes the TZ universal scheme. For every integer $k > 2$, the routing scheme has stretch factor $2k - 1$, assigns for each vertex a label of $o\left(k \cdot \log^2 n\right)$ bits, requires $\widetilde{\mathcal{O}}\left(n^{1/k}\right)$ bits for the routing table, requires $o\left(\log^2 n\right)$ bits for the message header and takes constant time on the routing decisions. Thorup and Zwick achieve a bound for routing tables sizes close to the optimal in this generalized version. Abraham et al. (2006b) prove that any routing scheme for graphs with arbitrary weights in the edges, arbitrary vertex labels and with stretch factor strictly less than $2k + 1$ requires $\Omega\left((n \cdot \log n)^{1/k}\right)$ bits for routing tables. For $k = 3$, for example, the generalized routing scheme with stretch factor $2k - 1 = 5$, which is strictly less than $2k + 1 = 7$, has routing tables of size $\widetilde{\mathcal{O}}\left(n^{1/3}\right)$ bits, which is an upper bound close to the lower bound $\Omega\left((n \cdot \log n)^{1/3}\right)$ presented by Abraham et al..

Besides these two schemes, Thorup and Zwick present two other compact routing schemes specialized in trees. The first one routes through shortest paths, assigns for each vertex a

label of $\mathcal{O}\left(\frac{\log^2 n}{\log\log n}\right)$ bits and takes constant time on the routing decisions. The routing algorithm uses only the labels of the local vertex and target, where the later is encoded in the message header. This scheme is **fixed-port**, that is, the port numbers assignment of each vertex is given by the network designer. The non-optimized TZ universal scheme, for example, is also fixed-port. Fraigniaud and Gavoille (2002) show that the bound achieved by Thorup and Zwick is optimal for the sum of label and routing table sizes in fixed-port routing schemes for trees that routes through shortest paths. On the other hand, a **designer-port** routing scheme has a preprocessing algorithm that also outputs a port number assignment. The designer of a designer-port routing scheme is free to assign port numbers aiming to optimize the labels and routing tables sizes. The second routing scheme for trees of Thorup and Zwick is designer-port, and has improved labels sizes of $(1 + o(1)) \cdot \log n$ bits. Fraigniaud and Gavoille (2001) also present independently both fixed-port and designer-port routing schemes for trees with results similar to Thorup and Zwick.

All cited routing schemes attach a label to the vertices with topological information about the underling network. The TZ universal scheme, for example, has labels that point to the closest center. Routing schemes that labels the vertices with any encoded topological information are **name-dependent**. In **name-independent** routing schemes, the network designer can label the vertices freely with any information that may not have relation to the network topology. Note that name-independence does not mean fixed-port, where on the first the *a priori* vertex labeling is allowed and on the second the *a priori* port number assignment is allowed. All network topological information is only distributed in the routing tables in name-independent routing schemes. These schemes are useful, for example, in applications that have intrinsic requirements on vertices names, such as locating nearby copies of replicated objects in DHP's (Abraham et al., 2004b), and in networks that constantly change their topology, for example tracking of mobile vertices (Awerbuch and Peleg, 1990).

Arias et al. (2003) present two name-independent routing schemes with stretch factor 5 for metric spaces. One requires $\widetilde{\mathcal{O}}\left(n^{1/2}\right)$ bits for routing tables and $\mathcal{O}\left(\log^2 n\right)$ bits for message header, and the other requires $\widetilde{\mathcal{O}}\left(n^{2/3}\right)$ bits for routing tables and $\mathcal{O}(\log n)$ bits for message header. Note that the first optimizes the routing tables sizes, while the second optimizes the messages sizes and, as a consequence, the number of bits transmitted per message in the network. Abraham et al. (2004a) present a name-independent routing scheme with improved stretch factor 3 that requires $\widetilde{\mathcal{O}}\left(n^{1/2}\right)$ bits for routing tables, $\mathcal{O}\left(\frac{\log^2 n}{\log\log n}\right)$ bits for message header and a routing decision that takes constant time. Name-independent routing through shortest paths in trees is a variant of the routing problem where is not possible to reduce the routing tables sizes to a polylogarithmic function. Laing and Rajaraman (2007) present a family of stars that requires $\Omega\left(n^{1/2}\right)$ bits for routing tables.

Metric embeddings, defined in Section 2.1.2, are also useful for designing routing schemes (Ban et al., 2010; Krioukov et al., 2009). However, Talwar (2004) claims that the embedding of some families of metric spaces into Euclidean spaces, for example, may not be isometric and may not have low distortions in the distances. He points to two properties of metric spaces that also defines routing tables and labels sizes of routing schemes for metric spaces, the **aspect ratio** and **doubling dimension**. The aspect ratio of the metric space (V, d) is defined as

$$\Delta = \frac{\max\limits_{i,j \in V}(d(i,j))}{\min\limits_{i,j \in V}(d(i,j))}.$$

The doubling dimension of a metric space is the smallest α such that any ball of radius r can be covered by 2^α balls of radius $r/2$. In the context of an undirected and weighted graph $G = (V, E)$ that defines a metric space (V, d), a ball of radius $r \geq 0$ positioned at vertex $u \in V$ is the set

$B(u) = \{v \in V | d(u,v) \leq r\}$, where $d(u,v)$ is the length of a minimum weighted path from u to v in G. Talwar presents a name-dependent routing scheme for metric spaces with doubling dimension α and aspect ratio Δ that has stretch factor $1+\varepsilon$, requires $\mathcal{O}\left(\alpha \cdot \left(\frac{6 \cdot \log \Delta}{\varepsilon \cdot \alpha}\right)^{\alpha} \cdot \log^2 \Delta\right)$ bits for routing tables, $\mathcal{O}(\alpha \cdot \log \Delta)$ bits for labels and $\mathcal{O}\left(\alpha \cdot \log^2 \Delta\right)$ bits for message header. Abraham et al. (2006a) present other alternative name-dependent routing scheme with stretch factor $1+\varepsilon$ that requires $\mathcal{O}\left((1/\varepsilon)^{\mathcal{O}(\alpha)} \cdot \log \Delta \cdot \log n\right)$ bits for routing tables, $\lceil \log n \rceil$ bits for labels and $\mathcal{O}(\log n)$ bits for message header. Note that "compact" is related to low values of doubling dimension in these routing schemes. More details about doubling dimension and metric spaces can be found in the book of Heinonen (2001).

The name-independent version of the routing schemes for metric spaces has a lower bound for stretch factor. Abraham et al. (2006a) prove that any name-independent compact (according to Definition 20) routing scheme for these graphs would have stretch factor of at least $3-\varepsilon$, for $\varepsilon \in [0,1)$. Konjevod et al. (2006) present, independently of Abraham et al., a name-independent compact routing scheme with stretch factor $9+\varepsilon$ that requires $\mathcal{O}\left((2+1/\varepsilon)^{\mathcal{O}(\alpha)} \cdot \log^2 \Delta \cdot \log n\right)$ bits for the routing tables and $\mathcal{O}(\log n)$ bits for the message header. One year latter, Konjevod et al. (2007b) present both, name-dependent and name-independent, routing schemes where the bounds for the routing tables, labels and headers are independent of the aspect ratio. Routing schemes with this property are known as **scale-free**. Their name-dependent scale-free routing scheme has stretch factor $1+\varepsilon$, requires $\mathcal{O}\left((1/\varepsilon)^{\mathcal{O}(\alpha)} \cdot \log^3 n\right)$ bits for the routing tables, $\lceil \log n \rceil$ bits for the labels and $\mathcal{O}\left(\frac{\log^2 n}{\log \log n}\right)$ bits for the message header. Their name-independent scale-free routing scheme has stretch factor $9+\varepsilon$, requires $\mathcal{O}\left((1/\varepsilon)^{\mathcal{O}(\alpha)} \cdot \log^3 n\right)$ bits for the routing tables and $\mathcal{O}\left(\frac{\log^2 n}{\log \log n}\right)$ bits for the message header. Note that the stretch factor bounds of the above name-independent routing schemes respect the bound of Abraham et al. (2006a). Besides that, low values for doubling dimension remains a desired property in all routing schemes for metric spaces.

Table 3.2 summarizes the main properties of each presented routing scheme. A routing scheme presented in each row, from left to right, is specialized in metric spaces (MS) or trees (T), is name-dependent (ND) or name-independent (NI) and has bounds for stretch factor, routing tables sizes in bits, vertices labels sizes in bits and message header size in bits.

3.3 Other Related Works

Some works contribute for the research of routing in small world networks from another points of view. Zhang et al. (2002) claim that the **Freenet's search algorithm** (Clarke et al., 2000) also finds paths with $\mathcal{O}\left(\log^2 n\right)$ expected size in the one-dimensional version of Kleinberg's model, where n is the number of vertices. Dietzfelbinger and Woelfel (2009) present a lower bound for Kleinberg's greedy routing algorithm in a small world model over a n-ring that generates q long-range edges in a particular way with an **arbitrary** probability distribution. They proved that their routing algorithm finds paths with $\Omega\left(\frac{\log^2 n}{q}\right)$ expected length. Liu et al. (2009) present a model that divides the $n \times n$ lattice into $k \times k$ clusters and generates **only one long-range edge per cluster** with the inverse second-power distribution. They present a greedy routing algorithm and prove that it finds paths with polylogarithmic expected length. Bakun and Konjevod (2010) present a routing algorithm that uses machine learning to decide, based on the successfully delivered messages, which edge to forward a received message. They execute the algorithm in synthetic and real small world networks and conclude experimentally that their algorithm outperforms the Kleinberg's greedy routing algorithm.

Table 3.2: Presented compact routing schemes and their properties.

Authors	Class	Name Indep.	Stretch	Routing Tables	Labels	Header
Thorup and Zwick (2001)	MS	ND	3	$\mathcal{O}\big((n\log n)^{1/2}\big)$	$(1{+}o(1))\log n$	$(1{+}o(1))\log n$
Thorup and Zwick (2001)	MS	ND	$2k{-}1$	$\widetilde{\mathcal{O}}\big(n^{1/k}\big)$	$o\big(k\log^2 n\big)$	$o\big(\log^2 n\big)$
Thorup and Zwick (2001)	T	ND	1	0	$\mathcal{O}\Big(\frac{\log^2 n}{\log\log n}\Big)$	$\mathcal{O}\Big(\frac{\log^2 n}{\log\log n}\Big)$
Thorup and Zwick (2001)	T	ND	1	0	$(1{+}o(1))\log n$	$(1{+}o(1))\log n$
Arias et al. (2003)	MS	NI	5	$\widetilde{\mathcal{O}}\big(n^{1/2}\big)$	0	$\mathcal{O}\big(\log^2 n\big)$
Arias et al. (2003)	MS	NI	5	$\widetilde{\mathcal{O}}\big(n^{2/3}\big)$	0	$\mathcal{O}(\log n)$
Abraham et al. (2004a)	MS	NI	3	$\widetilde{\mathcal{O}}\big(n^{1/2}\big)$	0	$\mathcal{O}\Big(\frac{\log^2 n}{\log\log n}\Big)$
Talwar (2004)	MS	ND	$1{+}\varepsilon$	$\mathcal{O}\big(\alpha\big(\frac{6\log\Delta}{\varepsilon\alpha}\big)^{\alpha}\log^2\Delta\big)$	$\mathcal{O}(\alpha\log\Delta)$	$\mathcal{O}\big(\alpha\log^2\Delta\big)$
Abraham et al. (2006a)	MS	ND	$1{+}\varepsilon$	$\mathcal{O}\big((1/\varepsilon)^{\mathcal{O}(\alpha)}\log\Delta\log n\big)$	$\lceil\log n\rceil$	$\mathcal{O}(\log n)$
Konjevod et al. (2006)	MS	NI	$9{+}\varepsilon$	$\mathcal{O}\big((2{+}1/\varepsilon)^{\mathcal{O}(\alpha)}\log^2\Delta\log n\big)$	0	$\mathcal{O}(\log n)$
Konjevod et al. (2007b)	MS	ND	$1{+}\varepsilon$	$\mathcal{O}\big((1/\varepsilon)^{\mathcal{O}(\alpha)}\log^3 n\big)$	$\lceil\log n\rceil$	$\mathcal{O}\Big(\frac{\log^2 n}{\log\log n}\Big)$
Konjevod et al. (2007b)	MS	NI	$9{+}\varepsilon$	$\mathcal{O}\big((1/\varepsilon)^{\mathcal{O}(\alpha)}\log^3 n\big)$	0	$\mathcal{O}\Big(\frac{\log^2 n}{\log\log n}\Big)$

There are also small world models that do not build the base graph over a lattice. Kleinberg (2001) presents another model that creates a complete k-ary tree such that the tree leaves are the vertex set V. He defines the distance function h between two vertices $u, v \in V$ as the height of the least common ancestor in the tree. The probability for generating an edge from u to v is proportional to a monotone non-increasing function in $h(u, v)$. Liben-Nowell et al. (2005) present a model that better represents a real social network called LiveJournal. They claim that the geographic distance g between two points $u, v \in V$ in a two-dimensional surface is not a suitable distance function to model this social network, where V is the vertex set. They define the **rank distance** from u to v as the number of vertices in V closer to u than v in the distance g. Their model generates an edge from u to v with probability proportional to the inverse of the rank distance from u to v rather than the inverse second-power distribution with g as distance function. Bringmann et al. (2017) point to the need of an underlying and well defined structure to execute the greedy routing in the small world models presented until the moment. They define a model that generates all edges at random and a greedy routing algorithm that succeeds with probability $\Omega(1)$.

Some works present distinct solutions in routing schemes with bounded stretch. Konjevod et al. (2007a) present two name-independent routing schemes for metric spaces with low doubling dimension, each with a particular feature. The first one has stretch factor $1 + \varepsilon$, which is impossible for name-independent routing schemes on those graphs, as Abraham et al. (2006a) prove, at the cost of a small fraction of vertices with routing tables of size $\mathcal{O}(n \cdot \log n)$ bits, and the remaining with routing tables with polylogarithmic number of bits, where $n = |V|$. The second scheme has polylogarithmic number of bits for the routing tables at the cost of a small fraction of source vertices with stretch factor $9 + \varepsilon$, and the remaining with stretch factor $1 + \varepsilon$. Brady and Cowen (2006) present a compact routing scheme for power-law graphs with **additive** stretch factor rather than multiplicative. They prove an additive stretch factor a using $\mathcal{O}\left(b \cdot \log^2 n\right)$ bits for the label and the routing table in each vertex, where $n = |V|$. They also claim through experiments that a and b assume low values for some power-law graphs. Chen et al. (2009) present a name-dependent routing scheme for power-law graphs with stretch factor 3 and routing tables with expected size of $\mathcal{O}\left(n^{\gamma} \cdot \log n\right)$ bits with high probability, where $n = |V|$, $\varepsilon < \gamma < 1/3 + \varepsilon$ and $\varepsilon > 0$. Tang et al. (2009) present, independently of Chen et al., a

name-dependent routing scheme for power-law graphs with stretch factor 3, and routing tables with size of $\widetilde{\mathcal{O}}\left(n^{1/3}\right)$ bits with high probability, where $n = |V|$. Some of the authors (Tang et al., 2013) also present a name-independent routing scheme with routing tables of expected size of $\widetilde{\mathcal{O}}\left(n^{1/2}\right)$ bits with high probability.

There are also experimental works that analyze the average of stretches and routing tables sizes of distinct routing schemes submitted to synthetic graphs and real-world networks. Krioukov et al. (2004) perform experiments with the optimized TZ universal scheme. They use random power-law graphs with the number of vertices between 10000 and 11000. The average of stretches is approximately 1.1, where up to 70% of all pairwise paths have minimum length, and the average of the routing tables sizes vary in 50 records. Strowes et al. (2011) evaluate the performance of the optimized TZ universal scheme and the compact routing scheme with additive stretch factor of Brady and Cowen (2006). They analyze the execution of both routing schemes on versions of the Internet Autonomous System graph sampled from a period of 14 years. They conclude that both schemes perform similarly on random power-law graphs in terms of the routing tables sizes and the average of stretches, as previous works suggest (Krioukov et al., 2004, 2007).

4 Small World on the Torus

This chapter presents a small world model based on Kleinberg's model, the problem of vertex labeling in graphs generated by the presented model, and a probabilistic analysis of the existence of cycles of size four that do not belong to the base two-dimensional square torus. The beginning of Section 3.1 presents the Kleinberg's model and the Section 2.2.2 presents the two-dimensional square torus and the torus distance.

The Section 4.1 presents the **undirected toroidal small world** (`UTSW`) model and bounds the normalizing factor in the model. It generates undirected graphs over the two-dimensional square torus using the inverse second-power distribution with the torus distance. Section 4.2 presents the formal definition of the toroidal small world labeling problem, which has a solution presented in Chapter 6, and also presents an analysis of the problem. Finally, Section 4.3 bounds the probability of the existence of four-cycles outside the two-dimensional square torus. This event occurs with small probability, which allows the using of the approach of detection of four-cycles to find a spanning torus.

This chapter refers to many concepts introduced in Chapter 2 because the contributions of the thesis are presented from here. Moreover, many of the results presented in this chapter are used in Chapter 6. The three sections of this chapter are the base of all results presented in Chapter 6. Thus, there are some common notations on both chapters. All the results presented in this chapter were published in the technical report of Viertel and Vignatti (2018a) and were submitted for publishing to the Theoretical Computer Science journal.

4.1 Undirected Toroidal Small World Model

This section presents the undirected toroidal small world (`UTSW`) model. The model is similar to the model of Kleinberg (2000a). Roughly, instead of using a grid, it "ties the borders" of the lattice, obtaining a torus. Also, Kleinberg uses directed long-range edges, but `UTSW` uses undirected instead. Let $S^2 = S \times S$ and $[\![n]\!] = \{0, 1, ..., n-1\}$. At first, the model generates a two-dimensional $n \times n$ undirected torus as follows. It creates n^2 vertices such that each vertex is a distinct element of $[\![n]\!]^2$. After that, each vertex (i, j) is connected with the vertices $(i, (j+1) \bmod n)$ and $((i+1) \bmod n, j)$ through undirected edges. This procedure generates a two-dimensional square torus of size n. This torus is denoted in the remaining of the thesis as $T = (V, E')$ of size n, unless the generated torus or T are locally redefined. Note that $|V| = n^2$ and $|E'| = 2|V|$. This thesis assumes sizes $n \geq 3$, otherwise the torus may have parallel edges or loops, both of which are not considered in this work.

Let d_{uv} be the distance between u and v on T, that is, the **torus distance**. Remembering that $d_{uv} = \min(|x_u - x_v|, n - |x_u - x_v|) + \min(|y_u - y_v|, n - |y_u - y_v|)$ for all $u, v \in V$, where (x_w, y_w) is defined for $w \in V$ during the torus generation. After the torus creation, each vertex $u \in V$ chooses another vertex $v \in V \setminus \{u\}$ to create an edge. The choice is sampled from the inverse second-power distribution, i.e., u chooses v with probability $Z_u \cdot d_{uv}^{-2}$, where Z_u is

the normalizing factor in u. The normalizing factor Z_u is a value that keeps the probability distribution summing to one in each u. Its value is given by

$$Z_u = \left(\sum_{w \in V \setminus \{u\}} d_{uw}^{-2} \right)^{-1}.$$

To avoid parallel edges, an edge is not created if it has already been created. The edges created in the torus generation process are **local edges** and those created by the randomized process are **long-range edges**. Let $\mathcal{C}_{uv}$ be the event of vertex u choosing the vertex v to create a long-range edge. Note that the probability distribution $\Pr(\mathcal{C}_{uv}) = Z_u \cdot d_{uv}^{-2}$, for all $v \in V \setminus \{u\}$, sums to one. This thesis also assumes that the positions (x_w, y_w) of all $w \in V$, computed during the torus generation, are not an output of the model. In fact, finding these positions is the problem solved in Chapter 6.

Theorem 27. *$Z_u = Z_v$ for all $u, v \in V$.*

Proof. Let $P_{ui} = \{w \in V | d_{uw} = i\}$ for all $1 \leq i \leq n$. The Figure 4.1 shows the vertices in P_{u3} in a torus of size $n = 7$. As T is symmetrical for all u and v, so $|P_{ui}| = |P_{vi}|$ and, then

$$Z_u = \left(\sum_{w \in V \setminus \{u\}} d_{uw}^{-2} \right)^{-1} = \left(\sum_{i=1}^{n} |P_{ui}| \cdot i^{-2} \right)^{-1} = \left(\sum_{i=1}^{n} |P_{vi}| \cdot i^{-2} \right)^{-1} = Z_v.$$

□

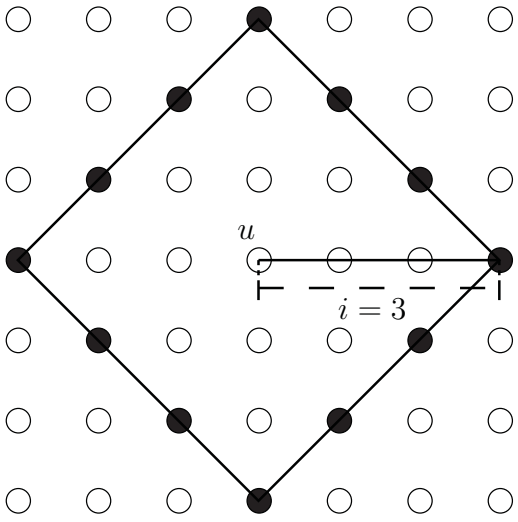


Figure 4.1: Vertices at a distance $i = 3 < n/2$ from u, with $n = 7$.

Despite each vertex having its own normalizing factor value, Theorem 27 shows that all of them are equal to each other. So, it is denoted Z, as shown in Definition 28. The `UTSW` model is formalized in Definition 29.

Definition 28. The **normalizing factor** is $Z = \left(\sum_{v \in V \setminus \{u\}} d_{uv}^{-2} \right)^{-1}$ for any $u \in V$.

Definition 29. The **undirected toroidal small world** (`UTSW`) model is a graph $G = (V, E)$ over a two-dimensional square torus of size $n \geq 3$ and, for all $u \in V$ and a $v \in V \setminus \{u\}$, the edge $\{u, v\}$ is included in E with probability $\Pr(\mathcal{C}_{uv}) = Z \cdot d_{uv}^{-2}$.

The remaining of this section assumes that G is a UTSW graph and presents some results related to the UTSW model that are used in Chapter 6.

Lemma 30. *Let $P_{ui} = \{v \in V | d_{uv} = i\}$ for each $u \in V$ and $1 \leq i \leq n$. Then, $|P_{ui}| = 4i$ if $i < n/2$, and $|P_{ui}| < 4i$ otherwise.*

Proof. The proof follows by "sweeping" in one of the two dimensions of T. When $i < n/2$,

$$|P_{ui}| = 1 + \sum_{j=1}^{i-1} 2 + 2 + \sum_{j=1}^{i-1} 2 + 1 = 4 + 2 \cdot 2(i-1) = 4i.$$

When $i \geq n/2$, $4i$ is an upper bound for $|P_{ui}|$. □

Fact 31. Let H_k be the k^{th} **harmonic number**, then $\ln(k+1) < H_k \leq \ln k + 1$.

Theorem 32. *The normalizing factor $Z < (4\ln(n/2))^{-1}$.*

Proof. Let $P_{ui} = \{v \in V | d_{uv} = i\}$ for any $u \in V$ and all $1 \leq i \leq n$. Given the Definition 28,

$$Z = \left(\sum_{v \in V \setminus \{u\}} d_{uv}^{-2} \right)^{-1} = \left(\sum_{i=1}^{n} |P_{ui}| \cdot i^{-2} \right)^{-1} < \left(\sum_{i=1}^{\lceil n/2 \rceil - 1} |P_{ui}| \cdot i^{-2} \right)^{-1} = (4H_{\lceil n/2 \rceil - 1})^{-1},$$

due to $i \leq \lceil n/2 \rceil - 1 < n/2$ and Lemma 30. By Fact 31 and $\lceil n/2 \rceil \geq n/2$, $Z < (4\ln(n/2))^{-1}$. □

Theorem 33. *The normalizing factor $Z > (4(\ln n + 1))^{-1}$.*

Proof. As at the beginning of the Theorem 32 proof and by Lemma 30,

$$Z > \left(\sum_{i=1}^{n} 4i \cdot i^{-2} \right)^{-1} = (4H_n)^{-1}.$$

By Fact 31, $Z > (4(\ln n + 1))^{-1}$. □

Theorem 33 implies that the upper bound of Z presented in Theorem 32 is tight.

4.2 Toroidal Small World Labeling Problem

A **position**, or **label**, $p = (p_1, p_2)$ in a two-dimensional square torus of size n is an element of $[\![n]\!]^2$, and p_i, for $i \in \{1, 2\}$, are the **coordinates** of p.

Definition 34. Let $d'_n : [\![n]\!]^2 \times [\![n]\!]^2 \to \mathbb{N}$ be the function such that $d'_n(x, y) = \sum_{i=1}^{2} \min(|x_i - y_i|, n - |x_i - y_i|)$, where $x, y \in [\![n]\!]^2$. A **two-dimensional toroidal vertex labeling function** of T is a function $\ell_T : V \to [\![n]\!]^2$ such that $d_{uv} = d'_n(\ell_T(u), \ell_T(v))$ for all $u, v \in V$ and $n = |V|^{1/2}$.

The function d' is similar to d, but d defines the distance between a pair of vertices on T and d' defines the distance between a pair of positions (labels). The notation ℓ_H corresponds to a vertex labeling function of the graph H in this section, rather than a norm function as used in Sections 2.1 and 2.2. The function ℓ_T defines a label to each vertex of T such that the distances d and d' between any pair of vertices of T are equal. The **toroidal small world labeling problem**

consists of finding a two-dimensional toroidal vertex labeling function $\ell_{T'}$ for a UTSW graph G and a spanning torus T' of size n. Note that T' is a spanning torus of G that may be distinct from the original torus T generated by UTSW model. This is an interesting problem because, given a two-dimensional toroidal vertex labeling function $\ell_{T'}$, it is possible to execute the Kleinberg's greedy routing algorithm for routing messages in G.

Note that G has at least one spanning two-dimensional torus, due to the Definition 29. Besides that, the long-range edges creation process in UTSW may generate others spanning two-dimensional tori, as explained next. The original torus T is composed only by local edges, while the others tori are composed by long-range edges also. The left side of the Figure 4.2 shows a part of a UTSW graph. In this case, each vertex chose a specific vertex to create a long-range edge, represented by the dashed arrows. This process generates a graph that may have more than one spanning torus. The Figure 4.2 highlights one of them in the right side.

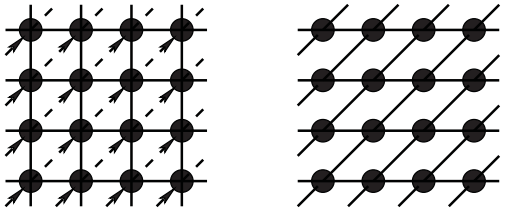


Figure 4.2: Graph with more than one spanning torus.

4.3 Cycles of Size Four Outside the Torus

The identification of a spanning two-dimensional torus is an important procedure. It can be used by a breadth-first search to label the vertices, as described in the Chapter 6. The used approach is to detect cycles of size four composed by distinct vertices, due to a two-dimensional square torus being an arrange of them.

A UTSW graph may have induced four-cycles composed by some long-range edges, instead of only local edges. The existence of this type of cycles is a problem in the torus identification process, if one uses the approach of searching for induced four-cycles. For any $u \in V$, there are only four cycles of size four rooted in u in the torus T. Besides that, the UTSW process may create long-range edges such that more induced four-cycles rooted in u appear in G. Despite this, Theorem 47 shows that the event of existence of four-cycles composed by at least one long-range edge occurs with small probability, and this is the main result of this section.

This section bounds the probability of the event that at least one four-cycle, rooted in u, is in G but not is in T. Such event, turns out to be a union of nine other events. Lemmas 38 to 46 show upper bounds on the probabilities of each of these nine events. The Lemmas 35 and 37 are technical results and, together with the Fact 36, are used to prove others results.

Lemma 35. $\sum_{v \in V \setminus \{u\}} d_{uv}^{-2} < 4(\ln n + 1)$ *for all* $u \in V$.

Proof. Grouping the terms of the sum by their values and using Lemma 30,

$$\sum_{v \in V \setminus \{u\}} d_{uv}^{-2} < \sum_{i=1}^{n} 4i \cdot i^{-2}.$$

By Fact 31, $\sum_{v \in V \setminus \{u\}} d_{uv}^{-2} < 4(\ln n + 1)$. □

Fact 36. The **Riemann zeta function** with parameter three is $\zeta(3) = \sum_{i=1}^{\infty} i^{-3} < 1,20206$.

Lemma 37. $\sum_{w \in V \setminus \{u,v\}} d_{uw}^{-2} d_{wv}^{-2} < 12\zeta(3) + 4$ *for all* $u, v \in V$ *and* $u \neq v$.

Proof. The distance $d_{uv} \leq n$ because T is a two-dimensional torus. Thus, the sum can be split into three sums:

$$\text{(i)} \quad \sum_{i=1}^{d_{uv}-1} \sum_{\substack{w \in V \setminus \{u,v\} \\ d_{uw}=i}} d_{uw}^{-2} d_{wv}^{-2},$$

$$\text{(ii)} \quad \sum_{\substack{w \in V \setminus \{u,v\} \\ d_{uw}=d_{uv}}} d_{uw}^{-2} d_{wv}^{-2} \text{ and}$$

$$\text{(iii)} \quad \sum_{i=d_{uv}+1}^{n} \sum_{\substack{w \in V \setminus \{u,v\} \\ d_{uw}=i}} d_{uw}^{-2} d_{wv}^{-2}.$$

For (i), $d_{uw} < d_{uv}$ for all terms. This fact allows the using of the triangle inequality property such that $d_{uv} - d_{uw}$ is positive for all $w \in V \setminus \{u, v\}$. So, this term is at most

$$\sum_{i=1}^{d_{uv}-1} \sum_{\substack{w \in V \setminus \{u,v\} \\ d_{uw}=i}} d_{uw}^{-2} (d_{uv} - d_{uw})^{-2}.$$

Since $d_{uw} = i$ and by Lemma 30,

$$\begin{aligned}
\sum_{i=1}^{d_{uv}-1} \sum_{\substack{w \in V \setminus \{u,v\} \\ d_{uw}=i}} d_{uw}^{-2} d_{wv}^{-2} &\leq 4 \sum_{i=1}^{d_{uv}-1} i^{-1} (d_{uv} - i)^{-2} \\
&= 4 \left(\sum_{i=1}^{\lfloor \frac{d_{uv}-1}{2} \rfloor} i^{-1} (d_{uv} - i)^{-2} + \sum_{i=1}^{\lceil \frac{d_{uv}-1}{2} \rceil} (d_{uv} - i)^{-1} i^{-2} \right) \\
&\leq 4 \left(\sum_{i=1}^{\lfloor \frac{d_{uv}-1}{2} \rfloor} i^{-1} i^{-2} + \sum_{i=1}^{\lceil \frac{d_{uv}-1}{2} \rceil} i^{-1} i^{-2} \right),
\end{aligned}$$

where the last inequality holds because $d_{uv} - i \geq i$ for all terms in both sums. As all sums terms are positive for $i \geq 1$, then

$$\sum_{i=1}^{d_{uv}-1} \sum_{\substack{w \in V \setminus \{u,v\} \\ d_{uw}=i}} d_{uw}^{-2} d_{wv}^{-2} < 8 \sum_{i=1}^{\infty} i^{-3} = 8\zeta(3), \text{ by Fact 36.}$$

For (ii), as $d_{wv} \geq 1$, $d_{uw} = d_{uv}$, $d_{uv} \geq 1$ and Lemma 30,

$$\sum_{\substack{w \in V \setminus \{u,v\} \\ d_{uw}=d_{uv}}} d_{uw}^{-2} d_{wv}^{-2} \leq \sum_{\substack{w \in V \setminus \{u,v\} \\ d_{uw}=d_{uv}}} d_{uw}^{-2} \leq 4 d_{uv} \cdot d_{uv}^{-2} \leq 4.$$

For (iii), the inequality

$$\sum_{i=d_{uv}+1}^{n} \sum_{\substack{w \in V \setminus \{u,v\} \\ d_{uw}=i}} d_{uw}^{-2} d_{wv}^{-2} \leq 4 \sum_{i=d_{uv}+1}^{n} i^{-1} (i - d_{uv})^{-2}$$

holds by a similar way of (i). The latter is equal to

$$4\sum_{i=1}^{n-d_{uv}}(d_{uv}+i)^{-1}i^{-2} < 4\sum_{i=1}^{n-d_{uv}} i^{-1}i^{-2} < 4\sum_{i=1}^{\infty} i^{-3} = 4\zeta(3), \text{ by Fact 36.}$$

Given the upper bounds on (i), (ii) and (iii), therefore $\sum_{w\in V\setminus\{u,v\}} d_{uw}^{-2}d_{wv}^{-2} < 12\zeta(3)+4$. □

Given $u \in V$, let $\mathcal{E}_u$ be the event of existence of at least one four-cycle, rooted in u, that is in G but not in T. Note that this event is equal to the event of UTSW creating long-range edges so that exists at least one four-cycle rooted in u that belongs to G composed by **at least one long-range edge**. Let s be a local edge and w be a long-range edge in a sequence of edges of a four-cycle rooted in u. Note that, there are 2^4 possible combinations of local and long-range edges and, one of them is certainly not in $\mathcal{E}_u$, which is (s,s,s,s), i.e. the four-cycle with only local edges. The others 15 combinations can be grouped in nine sub-events of $\mathcal{E}_u$, where each group leads to a different analysis, but the events belonging to a given group have the same analysis (which motivates the grouping). Then, $\mathcal{E}_u = \bigcup_{i=1}^{9}\mathcal{E}_{iu}$ and, each $\mathcal{E}_{iu}$ is defined by the following list.

- $\mathcal{E}_{1u}$: existence of at least one four-cycle, rooted in u, that belongs to G with edge sequence (s,s,s,w) or (w,s,s,s);
- $\mathcal{E}_{2u}$: existence of at least one four-cycle, rooted in u, that belongs to G with edge sequence (s,s,w,s) or (s,w,s,s);
- $\mathcal{E}_{3u}$: existence of at least one four-cycle, rooted in u, that belongs to G with edge sequence (s,s,w,w) or (w,w,s,s);
- $\mathcal{E}_{4u}$: existence of at least one four-cycle, rooted in u, that belongs to G with edge sequence (s,w,s,w) or (w,s,w,s);
- $\mathcal{E}_{5u}$: existence of at least one four-cycle, rooted in u, that belongs to G with edge sequence (s,w,w,s);
- $\mathcal{E}_{6u}$: existence of at least one four-cycle, rooted in u, that belongs to G with edge sequence (s,w,w,w) or (w,w,w,s);
- $\mathcal{E}_{7u}$: existence of at least one four-cycle, rooted in u, that belongs to G with edge sequence (w,s,s,w);
- $\mathcal{E}_{8u}$: existence of at least one four-cycle, rooted in u, that belongs to G with edge sequence (w,s,w,w) or (w,w,s,w);
- $\mathcal{E}_{9u}$: existence of at least one four-cycle, rooted in u, that belongs to G with edge sequence (w,w,w,w).

Lemmas 38 to 46 show upper bounds on the probabilities of each of these events. Finally, the Theorem 47 bounds the probability of $\mathcal{E}_u$. Remembering that $\mathcal{C}_{uv}$ is the event of vertex $u \in V$ choosing the vertex $v \in V \setminus \{u\}$ to create a long-range edge.

Lemma 38. $\Pr(\mathcal{E}_{1u}) < 2/3\ln^{-1}(n/2)$.

Proof. Let $A = \{a \in V | d_{ua} = 3\}$, then $\Pr(\mathcal{E}_{1u}) = \Pr\left(\bigcup_{a \in A} (\mathcal{C}_{ua} \cup \mathcal{C}_{au})\right)$. Using union bound, Definition 29, $d_{ua} = d_{au} = 3$ and $|A| \leq 4d_{ua} = 12$, by Lemma 30, then $\Pr(\mathcal{E}_{1u}) \leq 8/3Z$. By Theorem 32, $\Pr(\mathcal{E}_{1u}) < 2/3 \ln^{-1}(n/2)$. □

Lemma 39. $\Pr(\mathcal{E}_{2u}) < 64/9 \ln^{-1}(n/2)$.

Proof. Let $A = \{a \in V | d_{ua} = 1\}$ and $B = \{b \in V | d_{ub} = 2\}$, so $\Pr(\mathcal{E}_{2u}) = \Pr\left(\bigcup_{a \in A} \bigcup_{b \in B} (\mathcal{C}_{ab} \cup \mathcal{C}_{ba})\right)$. Using union bound, Definition 29 and $d_{ab} = d_{ba}$, then $\Pr(\mathcal{E}_{2u}) \leq 2Z \sum_{a \in A} \sum_{b \in B} d_{ab}^{-2}$. By Lemma 30, $|B| \leq 8$. When $|B| = 8$, $d_{ab} = 3$ occurs five times and $d_{ab} = 1$ occurs three times in the last sum. So, because $|A| = 4$, then $\Pr(\mathcal{E}_{2u}) \leq 256/9Z$. By Theorem 32, $\Pr(\mathcal{E}_{2u}) < 64/9 \ln^{-1}(n/2)$. □

Lemma 40. $\Pr(\mathcal{E}_{3u}) < (6\zeta(3) + 3/4) \ln^{-2}(n/2)$.

Proof. Let $A = \{a \in V | d_{ua} = 2\}$. The event $\mathcal{E}_{3u}$ is

$$\bigcup_{a \in A} \bigcup_{b \in V \setminus \{u,a\}} (\mathcal{C}_{ab} \cap \mathcal{C}_{bu}) \cup (\mathcal{C}_{ab} \cap \mathcal{C}_{ub}) \cup (\mathcal{C}_{ba} \cap \mathcal{C}_{ub}).$$

Note that the three intersections are between mutually independent events. So, in a similar way of Lemmas 38 and 39 proofs,

$$\Pr(\mathcal{E}_{3u}) \leq 3Z^2 \sum_{a \in A} \sum_{b \in V \setminus \{u,a\}} d_{ab}^{-2} d_{ub}^{-2}.$$

The last sum can be split into two other sums, one for $d_{ub} = 2$ and the other for $d_{ub} \geq 3$, because there is no $b \in V \setminus \{u, a\}$ such that $d_{ub} = 1$ in $\mathcal{E}_{3u}$. When $d_{ub} = 2$, $d_{ub} \leq d_{ab}$ for all $a \in A$ and, as a consequence, $\sum_{b \in V \setminus \{u,a\}} d_{ab}^{-2} d_{ub}^{-2} \leq 4 \cdot 2 \cdot 2^{-2} \cdot 2^{-2} = 1/2$, due to Lemma 30. When $d_{ub} \geq 3$, the triangle inequality property is used to find $d_{ab} \geq d_{ub} - 2$, because $d_{ua} = 2$. By this fact and using Lemma 30 and Fact 36,

$$\sum_{i=3}^{n} \sum_{\substack{b \in V \setminus \{u,a\} \\ d_{ub} = i}} d_{ab}^{-2} d_{ub}^{-2} \leq \sum_{i=3}^{n} 4i \cdot (i-2)^{-2} \cdot i^{-2} < 4 \sum_{i=1}^{\infty} i^{-3} = 4\zeta(3).$$

Therefore, $\Pr(\mathcal{E}_{3u}) < 3Z^2 \cdot 8(1/2 + 4\zeta(3))$, by the fact that $|A| \leq 4d_{ua} = 8$, due to Lemma 30. Using Theorem 32, $\Pr(\mathcal{E}_{3u}) < (6\zeta(3) + 3/4) \ln^{-2}(n/2)$. □

Lemma 41. $\Pr(\mathcal{E}_{4u}) < (16\zeta(3) + 5/4) \ln^{-2}(n/2)$.

Proof. Let $A = \{a \in V | d_{ua} = 1\}$ and, for a given $b \in V$, $C_b = \{c \in V | d_{bc} = 1\}$. Note that, for $x, y \in V$, $\mathcal{C}_{xy}$ is an event and C_b is a set. The event $\mathcal{E}_{4u}$ is

$$\bigcup_{a \in A} \bigcup_{b \in V \setminus \{u,a\}} \bigcup_{c \in C_b \setminus \{u,a\}} (\mathcal{C}_{ab} \cup \mathcal{C}_{ba}) \cap (\mathcal{C}_{uc} \cup \mathcal{C}_{cu}).$$

All simple events are mutually independent. Using the union bound, the simple events independence and Definition 29, then

$$\Pr(\mathcal{E}_{4u}) \leq 4Z^2 \sum_{a \in A} \sum_{b \in V \setminus \{u,a\}} \sum_{c \in C_b \setminus \{u,a\}} d_{ab}^{-2} d_{uc}^{-2}.$$

The second sum can be split for $d_{ab} = 2$ and $d_{ab} \geq 3$, because there is no $b \in V \setminus \{u, a\}$ such that $d_{ab} = 1$ for all $a \in A$ in $\mathcal{E}_{4u}$.

When $d_{ab} = 2$, $|\{b \in V \setminus \{u, a\} : d_{ab} = 2\}| \leq 4d_{ab} = 8$ holds, due to Lemma 30, for all $a \in A$. Note that, at most five of the at most eight vertices $b \in V \setminus \{u, a\}$ have their C_b with size $|\{c \in C_b \setminus \{u, a\} : d_{ab} = 2\}| \leq 4d_{bc} = 4$. The others at most three vertices do not belong to the event $\mathcal{E}_{4u}$. Furthermore, $d_{ab} \leq d_{uc}$ when $d_{ab} = 2$ and, then

$$\sum_{b \in V \setminus \{u,a\}} \sum_{c \in C_b \setminus \{u,a\}} d_{ab}^{-2} d_{uc}^{-2} \leq 5 \cdot 4 \cdot 2^{-2} \cdot 2^{-2} = 5/4.$$

When $d_{ab} \geq 3$, $d_{uc} \geq d_{ab} - 2$ for all $a \in A$. Combining these splittings of the sum for $d_{ab} = 2$ and $d_{ab} \geq 3$, together with Lemma 30,

$$\Pr(\mathcal{E}_{4u}) \leq 4Z^2 \cdot 4 \left(5/4 + 16 \sum_{i=3}^{n} i^{-1} \cdot (i-2)^{-2} \right).$$

As $\sum_{i=3}^{n} i^{-1} \cdot (i-2)^{-2} < \sum_{i=1}^{\infty} i^{-3}$, by Fact 36 and Theorem 32, $\Pr(\mathcal{E}_{4u}) < (16\zeta(3) + 5/4) \ln^{-2}(n/2)$. □

Lemma 42. $\Pr(\mathcal{E}_{5u}) < (9/2\zeta(3) + 63/128) \ln^{-2}(n/2)$.

Proof. Let $A = \{a \in V | d_{ua} = 1\}$ such that $A = \{a_1, a_2, a_3, a_4\}$ in any sequence. The event $\mathcal{E}_{5u}$ is

$$\bigcup_{\substack{a_i, a_j \in A \\ i<j}} \bigcup_{b \in V \setminus \{u, a_i, a_j\}} (\mathcal{C}_{a_i b} \cap \mathcal{C}_{b a_j}) \cup (\mathcal{C}_{a_i b} \cap \mathcal{C}_{a_j b}) \cup (\mathcal{C}_{b a_i} \cap \mathcal{C}_{a_j b}).$$

All simple events in the intersections are independent. So,

$$\Pr(\mathcal{E}_{5u}) \leq 3Z^2 \sum_{\substack{a_i, a_j \in A \\ i<j}} \sum_{b \in V \setminus \{u, a_i, a_j\}} d_{a_i b}^{-2} d_{a_j b}^{-2}.$$

Note that $d_{a_i b} \geq 2$ in $\mathcal{E}_{5u}$. When $d_{a_i b} = 2$, there is a $b \in V$ such that $b = a_j$ for all $a_i, a_j \in A$. Thus, $|\{b \in V \setminus \{u, a_i, a_j\} : d_{a_i b} = 2\}| \leq 4d_{a_i b} - 1 = 7$, due to Lemma 30. Furthermore, $d_{a_i b} \leq d_{a_j b}$ for all $a_i, a_j \in A$. Thus, the inequality

$$\sum_{b \in V \setminus \{u, a_i, a_j\}} d_{a_i b}^{-2} d_{a_j b}^{-2} \leq 7 \cdot 2^{-2} \cdot 2^{-2} = 7/16$$

holds when $d_{a_i b} = 2$. When $d_{a_i b} \geq 3$, $d_{a_j b} \geq d_{a_i b} - 2$ for all $a_i, a_j \in A$ and $b \in V \setminus \{u, a_i, a_j\}$. Combining with Lemma 30, the inequality

$$\sum_{b \in V \setminus \{u, a_i, a_j\}} d_{a_i b}^{-2} d_{a_j b}^{-2} \leq \sum_{k=3}^{n} 4k \cdot k^{-2} \cdot (k-2)^{-2}$$

holds. As $\sum_{k=3}^{n} k^{-1} \cdot (k-2)^{-2} < \sum_{k=1}^{\infty} k^{-3}$, then

$$\Pr(\mathcal{E}_{5u}) < 3Z^2 \sum_{\substack{a_i, a_j \in A \\ i<j}} \left(7/16 + 4 \sum_{k=1}^{\infty} k^{-3} \right).$$

By Lemma 30, $|A| = 4d_{ua} = 4$, thus there are $\binom{4}{2}$ distinct combinations of values for i and j. Therefore, $\Pr(\mathcal{E}_{5u}) < (9/2\zeta(3) + 63/128)\ln^{-2}(n/2)$, due to Fact 36 and Theorem 32. □

Lemma 43. $\Pr(\mathcal{E}_{6u}) < (12\zeta(3) + 4)(\ln n + 1)\ln^{-3}(n/2)$.

Proof. Let $A = \{a \in V | d_{ua} = 1\}$. The event $\mathcal{E}_{6u}$ is

$$\bigcup_{a\in A}\bigcup_{b\in V\setminus\{u,a\}}\bigcup_{c\in V\setminus\{u,a,b\}} \begin{array}{c}(\mathcal{C}_{ab}\cap\mathcal{C}_{bc}\cap\mathcal{C}_{uc})\cup(\mathcal{C}_{ab}\cap\mathcal{C}_{cb}\cap\mathcal{C}_{uc})\cup\\ (\mathcal{C}_{ba}\cap\mathcal{C}_{cb}\cap\mathcal{C}_{uc})\cup(\mathcal{C}_{ab}\cap\mathcal{C}_{bc}\cap\mathcal{C}_{cu}).\end{array}$$

All the intersections in $\mathcal{E}_{6u}$ are among mutually independent simple events. So, because $d_{ab} = d_{ba}$, $d_{bc} = d_{cb}$, $d_{uc} = d_{cu}$ and by Definition 29, the probabilities of the four combinations of intersections of simple events in $\mathcal{E}_{6u}$ are equal to $Z^3 \cdot d_{ab}^{-2} \cdot d_{bc}^{-2} \cdot d_{cu}^{-2}$, for all $a \in A$, $b \in V \setminus \{u, a\}$ and $c \in V \setminus \{u, a, b\}$. Using the union bound,

$$\Pr(\mathcal{E}_{6u}) \leq 4Z^3 \sum_{a\in A}\sum_{b\in V\setminus\{u,a\}} d_{ab}^{-2} \sum_{c\in V\setminus\{u,a,b\}} d_{bc}^{-2} d_{cu}^{-2}.$$

Using Lemmas 37 and 35 and due to $|A| = 4d_{ua} = 4$, by Lemma 30 and Theorem 32, $\Pr(\mathcal{E}_{6u}) < (12\zeta(3) + 4)(\ln n + 1)\ln^{-3}(n/2)$. □

Lemma 44. $\Pr(\mathcal{E}_{7u}) < (6\zeta(3) + 21/32)\ln^{-2}(n/2)$.

Proof. For a given $a \in V$, let $B_a = \{b \in V | d_{ab} = 2\}$. The event $\mathcal{E}_{7u}$ is

$$\bigcup_{a\in V\setminus\{u\}}\bigcup_{b\in B_a\setminus\{u\}} (\mathcal{C}_{ua}\cap\mathcal{C}_{bu})\cup(\mathcal{C}_{au}\cap\mathcal{C}_{ub})\cup(\mathcal{C}_{au}\cap\mathcal{C}_{bu}).$$

All intersections are between mutually independent simple events. Using the union bound, the events independence, Definition 29, $d_{au} = d_{ua}$ and $d_{bu} = d_{ub}$, then

$$\Pr(\mathcal{E}_{7u}) \leq 3Z^2 \sum_{a\in V\setminus\{u\}}\sum_{b\in B_a\setminus\{u\}} d_{ua}^{-2} d_{ub}^{-2}.$$

Note that $d_{ua} \geq 2$ in $\mathcal{E}_{7u}$. So, the first sum can be split for $d_{ua} = 2$ and $d_{ua} \geq 3$.

When $d_{ua} = 2$, there is a $b \in B_a$ such that $b = u$ for all $a \in V \setminus \{u\}$. As a consequence, the inequalities $|\{b \in B_a \setminus \{u\} : d_{ua} = 2\}| \leq 4d_{ab} - 1 = 7$ and $|\{a \in V \setminus \{u\} : d_{ua} = 2\}| \leq 4d_{ua} = 8$ hold, due to Lemma 30. Besides that, $d_{ua} \leq d_{ub}$ for all $a \in V \setminus \{u\}$ and $b \in B_a \setminus \{u\}$. Thus,

$$\sum_{a\in V\setminus\{u\}}\sum_{b\in B_a\setminus\{u\}} d_{ua}^{-2} d_{ub}^{-2} \leq 8 \cdot 7 \cdot 2^{-2} \cdot 2^{-2} = 7/2 \text{ when } d_{ua} = 2.$$

When $d_{ua} \geq 3$, then $d_{ub} \geq d_{ua} - 2$ for all $a \in V \setminus \{u\}$ and $b \in B_a \setminus \{u\}$. Therefore,

$$\sum_{a\in V\setminus\{u\}}\sum_{b\in B_a\setminus\{u\}} d_{ua}^{-2} d_{ub}^{-2} \leq 32\sum_{i=3}^{n} i^{-1} \cdot (i-2)^{-2},$$

due to the facts $|\{a \in V \setminus \{u\} : d_{ua} \geq 3\}| \leq 4d_{ua}$ and $|\{b \in B_a \setminus \{u\} : d_{ua} \geq 3\}| \leq 4d_{ab} = 8$, by Lemma 30. Combining the cases for $d_{ua} = 2$ and $d_{ua} \geq 3$, then

$$\Pr(\mathcal{E}_{7u}) \leq 3Z^2\left(7/2 + 32\sum_{i=3}^{n} i^{-1} \cdot (i-2)^{-2}\right).$$

As $\sum_{i=3}^{n} i^{-1} \cdot (i-2)^{-2} < \sum_{i=1}^{\infty} i^{-3}$ and by Fact 36 and Theorem 32, so $\Pr(\mathcal{E}_{7u}) < (6\zeta(3) + 21/32)\ln^{-2}(n/2)$. □

Lemma 45. $\Pr(\mathcal{E}_{8u}) < (12\zeta(3)+4)(\ln n+1)\ln^{-3}(n/2)$.

Proof. For a given $a \in V$, let $B_a = \{b \in V | d_{ab} = 1\}$. The event $\mathcal{E}_{8u}$ is

$$\bigcup_{a\in V\setminus\{u\}} \bigcup_{b\in B_a\setminus\{u\}} \bigcup_{c\in V\setminus\{u,a,b\}} \begin{array}{l}(\mathcal{C}_{ua}\cap\mathcal{C}_{bc}\cap\mathcal{C}_{cu})\cup(\mathcal{C}_{au}\cap\mathcal{C}_{bc}\cap\mathcal{C}_{uc})\cup \\ (\mathcal{C}_{au}\cap\mathcal{C}_{bc}\cap\mathcal{C}_{cu})\cup(\mathcal{C}_{au}\cap\mathcal{C}_{cb}\cap\mathcal{C}_{uc}).\end{array}$$

The proof follows similarly to the proof of Lemma 43. □

Lemma 46. $\Pr(\mathcal{E}_{9u}) < (3/4\zeta(3)+1/4)(\ln n+1)^2\ln^{-4}(n/2)$.

Proof. The event $\mathcal{E}_{9u}$ is

$$\bigcup_{a\in V\setminus\{u\}} \bigcup_{b\in V\setminus\{u,a\}} \bigcup_{c\in V\setminus\{u,a,b\}} (\mathcal{C}_{ua}\cap\mathcal{C}_{ab}\cap\mathcal{C}_{bc}\cap\mathcal{C}_{cu}).$$

All simple events in the intersections are mutually independent. Using the union bound, the independence of the simple events and Definition 29,

$$\Pr(\mathcal{E}_{9u}) \le Z^4 \sum_{a\in V\setminus\{u\}} d_{ua}^{-2} \sum_{b\in V\setminus\{u,a\}} d_{ab}^{-2} \sum_{c\in V\setminus\{u,a,b\}} d_{bc}^{-2} d_{cu}^{-2}.$$

By Lemmas 37 and 35 and Theorem 32, $\Pr(\mathcal{E}_{9u}) < (3/4\zeta(3)+1/4)(\ln n+1)^2\ln^{-4}(n/2)$. □

Theorem 47. $\Pr(\mathcal{E}_u) = \mathcal{O}\left(\log^{-1} n\right)$.

Proof. The bound for the probability of the event $\mathcal{E}_u = \bigcup_{i=1}^{9} \mathcal{E}_{iu}$ follows by the union bound and Lemmas 38 to 46. Then,

$$\begin{aligned}\Pr(\mathcal{E}_u) &\le \sum_{i=1}^{9} \Pr(\mathcal{E}_{iu}) \\ &< 70/9\ln^{-1}(n/2) + \\ &\quad (65/2\zeta(3)+403/128)\ln^{-2}(n/2) + \\ &\quad (24\zeta(3)+8)(\ln n+1)\ln^{-3}(n/2) + \\ &\quad (3/4\zeta(3)+1/4)(\ln n+1)^2\ln^{-4}(n/2).\end{aligned}$$

As $(\ln n+1)/\ln(n/2)$ is $\mathcal{O}(1)$, so $\Pr(\mathcal{E}_u)$ is $\mathcal{O}\left(\log^{-1} n\right)$. □

Theorem 47 means that the event $\mathcal{E}_u$ of existence of at least one four-cycle, rooted in a vertex $u \in V$, that is in G but not in T occurs with small probability, according to Definition 25 and stated in Corollary 48. Chapter 6 presents the labeling algorithm, that uses the results of this section.

Corollary 48. *$\mathcal{E}_u$ occurs **with small probability**.*

Proof.

$$\begin{aligned}\lim_{n\to\infty} \Pr(\mathcal{E}_u) &\le \lim_{n\to\infty} \lambda \cdot \log^{-1} n \\ &= 0.\end{aligned}$$

□

5 Small World on the Octahedron and Spheres

This chapter uses Facts 31 and 36, both presented in Chapter 4. The events $\mathcal{C}_{uv}$, $\mathcal{E}_{iu}$ and $\mathcal{E}_u$ and the distance function d are redefined here. Many related works (Manku et al., 2004; Martel and Nguyen, 2004; Zeng et al., 2005; Liu et al., 2009) present models that build a base graph over bounded geometries that are plane in the three-dimensional euclidean space. The model of Kleinberg (2000a) is an example that generates a clustered network over the square with vertices $(1, 1, 0)$, $(1, n, 0)$, $(n, n, 0)$ and $(n, 1, 0)$. The model of Martel and Nguyen (2004) uses the two-dimensional square torus, which is a solid geometry with genus one. There is a lack of small world models that generate the base graph over genus-zero solid geometries, that is, are homeomorphic to spheres, such as, for example, any Platonic solid.

The Section 5.2 presents the octahedral small world model (`OSW`) that generates the base graph over the octahedron. This section also defines a greedy routing algorithm that finds paths with $\mathcal{O}\left(\log^2 n\right)$ expected size in octahedral small world graphs. Section 5.1.1 relates the base graph with spheres, providing a connection with, for example, social networks over the Earth globe. The vertices on the octahedron can be projected on the surface of spheres. This section relates the distances between neighbor vertices on both geometry, while the size n of the octahedron and the sphere radius go to infinity. As in the octahedron, where the distances are bounded by a constant for increasing n, there are spheres where the distances are also bounded by a constant for increasing radius. Then, there is a family of spheres that are asymptotically similar to the octahedron in terms of distances on the surface. Section 5.3 proves that the expected number of cycles of size three in these graphs is $\Theta\left(n^2\right)$. Theorems 64 and 66 can be used, for example, in the analysis of running time of algorithms that execute searches of three-cycles. Identifying the base graph by performing a local search for the cycles in each vertex is an approach used in Chapter 6. All results presented in this chapter were published in the technical report of Viertel and Vignatti (2018b) and were submitted for publishing to the Information Processing Letters journal.

5.1 Octahedrons and Graphs

Let the r**-octahedron** be the set $O_r = \left\{u \in \mathbb{R}^3 : \sum_{i=1}^{3} |u_i| = r\right\}$, with $r > 0$ and $u = (u_1, u_2, u_3)$. Let $V = O_n \cap \mathbb{Z}^3$ for $n \in \{1, 2, 3, ...\}$, i.e., V is the set of all three-dimensional vectors with integer coordinates in the n-octahedron. There exists an undirected graph on V that "wraps" O_n. The edges do not pass through the "inside" of O_n, linking only the closest vectors on O_n. So, each $v \in V$ has an undirected edge to all $w \in V \setminus \{v\}$ such that $|v_i - w_i| \leq 1$, for $1 \leq i \leq 3$. Definition 49 formalizes the n-octahedral graph. Figure 5.1 illustrates a two-octahedral graph.

Lemma 50 shows the number of vertices of G'_n. Note that, n is the size of the n-octahedral graph and $|V|$ is the number of vertices. Lemma 51 shows the number of edges of G'_n.

Definition 49. The **n-octahedral graph** with size $n \in \{1, 2, 3, ...\}$ is $G'_n = (V, E')$ such that

$$V = \left\{ u \in \mathbb{Z}^3 : \sum_{i=1}^{3} |u_i| = n \right\} \text{ and } E' = \{\{v, w\} \subset V : |v_i - w_i| \leq 1, \forall 1 \leq i \leq 3\}.$$

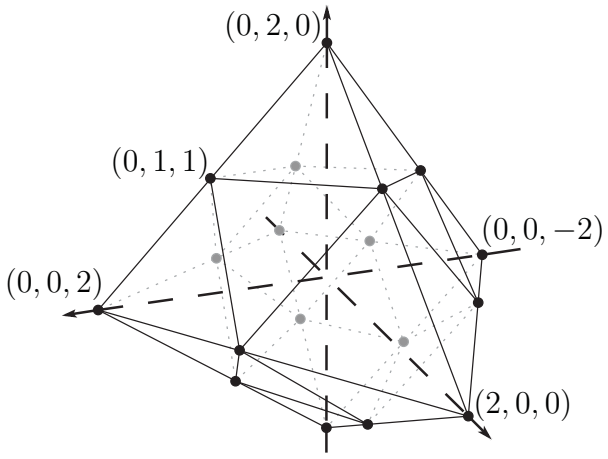


Figure 5.1: A two-octahedral graph.

Lemma 50. $|V| = 4n^2 + 2$.

Proof. The proof follows counting all combinations of u_1, u_2 and u_3 that satisfy the Definition 49. For $u_1 = -n$ and $u_1 = n$, $u_2 = u_3 = 0$, that are two combinations. For each $-n+1 \leq u_1 \leq n-1$, for $u_2 = -(n - |u_1|)$ and $u_2 = n - |u_1|$, $u_3 = 0$, that are two combinations for each u_1. For each $-n + 1 \leq u_1 \leq n - 1$ and $-(n - |u_1|) + 1 \leq u_2 \leq (n - |u_1|) - 1$, $u_3 = -(n - |u_1| - |u_2|)$ and $u_3 = n - |u_1| - |u_2|$, that are two combinations for each u_1 and u_2. Then,

$$|V| = 2 + \left(\sum_{u_1=-n+1}^{n-1} 2 \right) + \left(\sum_{u_1=-n+1}^{n-1} \left(\sum_{u_2=-(n-|u_1|)+1}^{(n-|u_1|)-1} 2 \right) \right) = 4n^2 + 2.$$

□

Lemma 51. $|E'| = 12n^2$.

Proof. By Definition 49, for all $u \in V$, the two neighbors of u with the same value of the third coordinate, when $-n + 1 \leq u_3 \leq n - 1$, are:

- $(-1, u_2 - 1, u_3) \in V$ and $(1, u_2 - 1, u_3) \in V$ if $u_1 = 0$ and $u_2 > 0$;
- $(-1, u_2 + 1, u_3) \in V$ and $(1, u_2 + 1, u_3) \in V$ if $u_1 = 0$ and $u_2 < 0$;
- $(u_1 - 1, -1, u_3) \in V$ and $(u_1 - 1, 1, u_3) \in V$ if $u_1 > 0$ and $u_2 = 0$;
- $(u_1 + 1, -1, u_3) \in V$ and $(u_1 + 1, 1, u_3) \in V$ if $u_1 < 0$ and $u_2 = 0$;

- $(u_1 - 1, u_2 + 1, u_3) \in V$ and $(u_1 + 1, u_2 - 1, u_3) \in V$ if $u_1 > 0$ and $u_2 > 0$ or $u_1 < 0$ and $u_2 < 0$;
- $(u_1 - 1, u_2 - 1, u_3) \in V$ and $(u_1 + 1, u_2 + 1, u_3) \in V$ if $u_1 < 0$ and $u_2 > 0$ or $u_1 > 0$ and $u_2 < 0$.

This arrangement of edges defines a sequence of cycles with increasing sizes for u_3 growing from $-n + 1$ to zero and decreasing sizes from one to $n - 1$, as Figure 5.2 shows. There are two similar sequences of cycles for u_1 and u_2. Counting one edge for each vertex of each cycle in each coordinate,

$$|E'| = \sum_{\{i,j\} \in \binom{\{1,2,3\}}{2}} \left(\left(\sum_{u_i=-n+1}^{n-1} 2 \right) + \left(\sum_{u_i=-n+1}^{n-1} \left(\sum_{u_j=-(n-|u_i|)+1}^{(n-|u_i|)-1} 2 \right) \right) \right) = 12n^2.$$

□

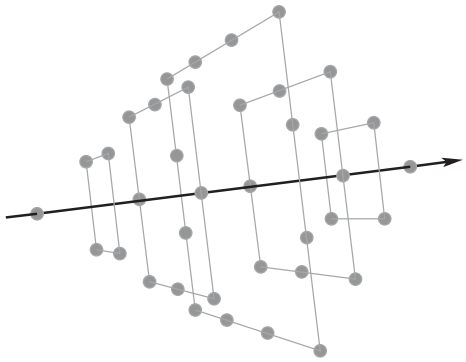


Figure 5.2: The underlying cycles surrounding an axis.

5.1.1 The n-Octahedral Graph and Spheres

Let the distance between a pair $\{u, v\} \in E'$ on O_n be the length in the euclidean norm of the line segment bounded by $u \in V$ and $v \in V$. Let the distance between u and v on the sphere with radius $r > 0$ be the length in the euclidean space of the smallest circular arc defined by the projections of u and v on the surface of the sphere with radius r and center at $(0, 0, 0)$. Figure 5.3 shows the projections of u and v, the circular arc defined by them and the angle α_{uv} between u and v in radians. The **great-circle distance** is defined by the well known relation

$$\frac{d_{uv}^r}{2\pi r} = \frac{\alpha_{uv}}{2\pi}.$$

Thus $d_{uv}^r = r \cdot \alpha_{uv}$ for all $\{u, v\} \in E'$, where d_{uv}^r is the distance between u and v on the sphere with radius r.

By Definition 49, the distance on O_n between u and v is $\sqrt{2}$ for all $\{u, v\} \in E'$ and $n \geq 1$. On the other hand, the distances on the sphere d_{uv}^r vary according to α_{uv} for all radius $r > 0$. In the case of $r = n$, d_{uv}^n diverge as n goes to infinity, because r increases faster than α_{uv} inversely decrease. On the other extreme, when r is a positive constant λ', $d_{uv}^{\lambda'}$ converge to 0 as n goes to infinity. Theorem 52 shows that all d_{uv}^r are in the interval $(0, \lambda]$ when $r \leq \lambda \left(2 \arctan\left(\frac{\sqrt{6}}{2n}\right)\right)^{-1}$ for a positive constant λ. Therefore, the distances on the sphere are

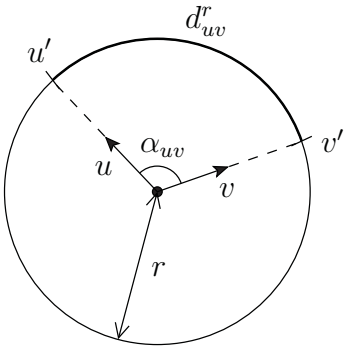


Figure 5.3: Distance between the projections of u and v, u' and v' respectively, on a sphere with radius $r > 0$.

$\mathcal{O}(1)$ for some values of r, which is asymptotically similar to the distances on O_n. Figure 5.4 shows the radius upper bound function plot for $\lambda = 1$.

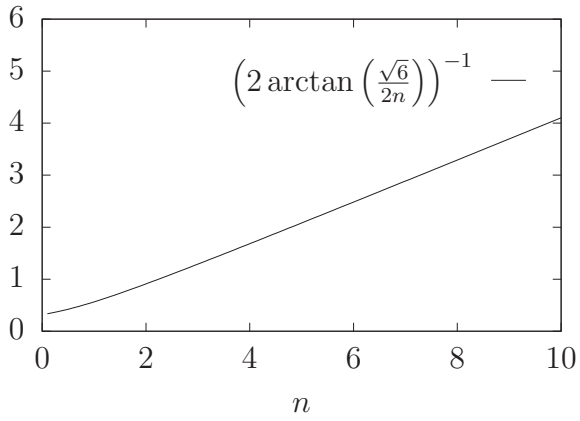


Figure 5.4: The upper bound function of $r > 0$.

Theorem 52. *The distances on all spheres with radius* $0 < r \leq \lambda \left(2\arctan\left(\frac{\sqrt{6}}{2n}\right)\right)^{-1}$ *are* $0 < d^r_{uv} \leq \lambda$*, for all* $\{u, v\} \in E'$*,* $n \geq 1$ *and constants* $\lambda > 0$*.*

Proof. For $d^r_{uv} \leq \lambda$ for all $\{u, v\} \in E'$, $r \leq \lambda \cdot \alpha^{-1}_{uv}$. The analysis follows on all planes defined by all u, v and $(0, 0, 0)$ as shown in Figure 5.5. Given the legs a and b of the two right triangles defined in each plane, $\alpha_{uv} = \arctan(a/b) + \arctan\left(\left(\sqrt{2} - a\right)/b\right)$. The values of all α_{uv} are maximum for $\frac{\partial \alpha_{uv}}{\partial a} = 0$, that is $a = \sqrt{2}/2$. Moreover, α_{uv} is upper bounded when b is minimum because a is constant, for all u and v. The minimum value of b is the radius of the inscribed sphere in O_n, that is $\left(\sqrt{3}/3\right) n$. Then $\alpha_{uv} \leq 2\arctan\left(\frac{\sqrt{6}}{2n}\right)$ for all u and v and $r \leq \lambda \left(2\arctan\left(\frac{\sqrt{6}}{2n}\right)\right)^{-1} \leq \lambda \cdot \alpha^{-1}_{uv}$. □

5.2 Octahedral Small World Model

Let $d : V \times V \to \mathbb{N}$ be the distance function defined by the length of a minimum path between all pairs of vertices $u, v \in V$ in G'_n. A **path** is a sequence of distinct vertices which each two consecutive vertices the first is incident to the second. The **length** of a path is the number of vertices of the path minus one. A **minimum path** is a path with the minimum length among all

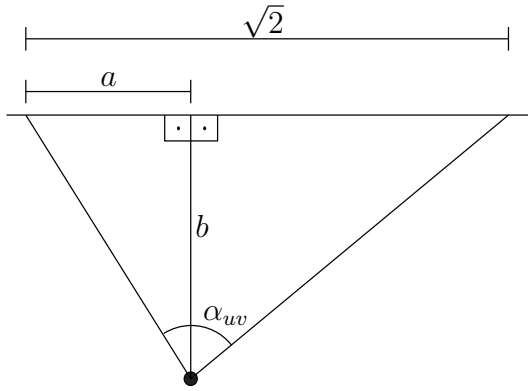


Figure 5.5: Right triangles defined by u, v and O_n.

paths. Let $\mathcal{C}_{uv}$ be the event of $u \in V$ choosing $v \in V \setminus \{u\}$ to create the directed edge $(u, v) \in E$ and

$$Z_u = \left(\sum_{w \in V \setminus \{u\}} d_{uw}^{-2} \right)^{-1}$$

be the **normalizing factor** of all u.

Definition 53. The **octahedral small world** (OSW) model is $G_n = (V, E)$ such that for the n-octahedral graph $G'_n = (V, E')$:

1. for all $\{u, v\} \in E'$, both directed edges (u, v) and (v, u) are included in E and;
2. for all $u \in V$ and a $v \in V \setminus \{u\}$, (u, v) is included in E with probability $\Pr(\mathcal{C}_{uv}) = Z_u \cdot d_{uv}^{-2}$.

Note that G'_n is undirected, as shown in Definition 49, and G_n is directed, as shown in Definition 53. All undirected edges $\{u, v\} \in E'$ correspond to the pair of directed edges $(u, v), (v, u) \in E$. The **long-range edges** are those generated by the independent random trials of OSW. Follows a greedy routing algorithm that finds small paths in G_n. Given a vertex $u \in V$ and a message, the algorithm sends the message to the vertex $v \in \mathcal{N}^+_{G_n}(u)$ with minimum angle α_{vt} with the target vertex $t \in V$, where $\mathcal{N}^+_{G_n}(u)$ is the set of out-neighbors of u in G_n. The algorithm selects v computing

$$\underset{v \in \mathcal{N}^+_{G_n}(u)}{\operatorname{argmax}} \left(\frac{v \cdot t}{|v||t|} \right), \text{ where } v \cdot t = \sum_{i=1}^{3} v_i \cdot t_i, \; |w| = \left(\sum_{i=1}^{3} w_i^2 \right)^{1/2}$$

and t is in the message header.

Theorem 56 proves that this routing algorithm finds paths with squared logarithmic length in n. The proof strategy is inspired in the work of Kleinberg (2000a). Lemma 54 proves the upper bound of Z_u for all $u \in V$, which is used throughout Section 5.3. Lemma 55 proves the lower bound of Z_u for all u, which is used in the proof of Theorem 56. Let $P_{ui} = \{w \in V | d_{uw} = i\}$ for distances $i \geq 1$. Figure 5.6 shows the vertices in P_{u1}, P_{u2} and P_{u3} in the bold cycles, where u is the central vertex.

Lemma 54. *The normalizing factor* $Z_u < \ln^{-1}(n + 1)$.

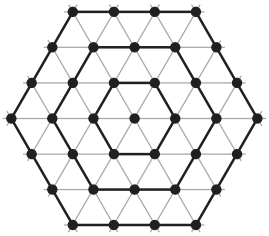


Figure 5.6: Sets of vertices at the distances one, two and three from the central vertex.

Proof. The longest distance in G'_n is at most $2n$. Moreover, $|P_{ui}| \geq i + 1 > i$ for $i \leq n$, as shown in the bold area of the Figure 5.7, where $u = t$. As $\sum_{i=1}^{n} i^{-1} > \ln(n+1)$, then

$$Z_u = \left(\sum_{i=1}^{2n} |P_{ui}| \cdot i^{-2} \right)^{-1} < \left(\sum_{i=1}^{n} |P_{ui}| \cdot i^{-2} \right)^{-1} < \left(\sum_{i=1}^{n} i^{-1} \right)^{-1} < \ln^{-1}(n+1).$$

□

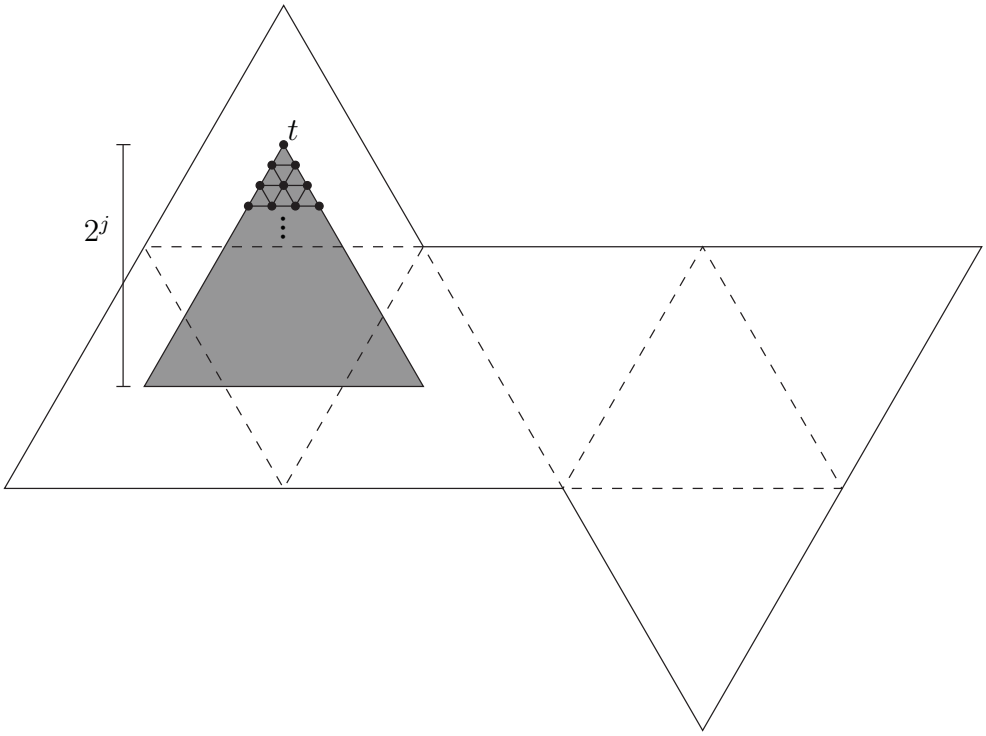


Figure 5.7: Some vertices in B_{tj} with distance of at most $2^j \leq n$ from $t \in V$ in an octahedron folding.

Lemma 55. *The normalizing factor* $Z_u \geq (6 \ln(2en))^{-1}$.

Proof. As $|P_{ui}| \leq 6i$ and $\sum_{i=1}^{m} i^{-1} \leq \ln(m) + 1$, then

$$Z_u = \left(\sum_{i=1}^{2n} |P_{ui}| \cdot i^{-2} \right)^{-1} \geq \left(\sum_{i=1}^{2n} 6i \cdot i^{-2} \right)^{-1} \geq (6(\ln(2n) + 1))^{-1} = (6 \ln(2en))^{-1}.$$

□

Theorem 56. *The greedy routing algorithm performs* $\mathcal{O}\left(\log^2 n\right)$ *expected number of forwardings.*

Proof. Let the subdivision of the path with source $s \in V$ and target $t \in V$ that the greedy routing algorithm finds be such that the algorithm is in phase $j \geq 1$ if $2^j < d_{ut} \leq 2^{j+1}$, where $u \in V$ is a vertex of the path, and is in phase zero when $d_{ut} \leq 2$. The phase j ends if the current vertex u with the message has a long-range edge to any vertex $v \in B_{tj}$, where $B_{tj} = \{w \in V | d_{wt} \leq 2^j\}$. By Definition 53 and Lemma 55, $\Pr(\mathcal{C}_{uv}) = Z_u \cdot d_{uv}^{-2} \geq (6 \ln(2en) d_{uv}^2)^{-1}$. As there are at least $\sum_{j=1}^{2^j} j$ vertices $w \in V$ such that $d_{wt} \leq 2^j$ (in the bold area in Figure 5.7), then $|B_{tj}| > 2^{2j-1}$. Besides, $d_{uv} \leq 2^{j+1} + 2^j < 2^{j+2}$ for all u and v. As all $\mathcal{C}_{uv}$ are disjoint, then the probability of the phase j ends in u is

$$\Pr\left(\bigcup_{v \in B_{tj}} \mathcal{C}_{uv}\right) = \sum_{v \in B_{tj}} \Pr(\mathcal{C}_{uv}) > \frac{2^{2j-1}}{6 \ln(2en)(2^{j+2})^2} = (192 \ln(2en))^{-1}.$$

Let X_j be the random variables that count the number of forwardings in the phases $j \geq 1$. As X_j are geometric random variables, then $\mathrm{E}[X_j] < 192 \ln(2en)$. Let X be the random variable that counts the total number of forwardings in the greedy routing. As $d_{st} \leq 2n$, then $X \leq 2 + \sum_{j=1}^{\lceil \log n \rceil + 1} X_j$ and $\mathrm{E}[X] < (\lceil \log n \rceil + 1)(192 \ln(2en)) + 2$ by the linearity of expectation. □

5.3 Three-Cycles not in the n-Octahedral Graph

The long-range edges generation in `OSW` may create new three-cycles that do not belong to the base n-octahedral graph G'_n. The n-octahedral graph is a well structured arrangement of three-cycles and the long-range edges "hide" it in G_n. A three-cycle in G_n and not in G'_n has at least one long-range edge. Besides, a three-cycle is a sequence of three edges where a long-range edge may assume any position. Considering these, the fact that G_n is directed and $u \in V$ is the first vertex of the three-cycle, there are seven distinct compositions of edges that define three-cycles rooted in u, in G_n and not in G'_n. Let s be a directed edge in G'_n and w be a long-range edge generated in `OSW`. These compositions are represented by the events $\mathcal{E}_{iu}$. All events $\mathcal{E}_{iu}$ refer to the existence of at least one three-cycle, rooted in u, but each specific event having an edge sequence as follows: $\mathcal{E}_{1u}$ with edge sequence (s, s, w); $\mathcal{E}_{2u}$ with edge sequence (s, w, s); $\mathcal{E}_{3u}$ with edge sequence (s, w, w); $\mathcal{E}_{4u}$ with edge sequence (w, s, s); $\mathcal{E}_{5u}$ with edge sequence (w, s, w); $\mathcal{E}_{6u}$ with edge sequence (w, w, s); $\mathcal{E}_{7u}$ with edge sequence (w, w, w).

Then, the event of the existence of at least one three-cycle in G_n, not in G'_n and rooted in u is $\mathcal{E}_u = \bigcup_{i=1}^{7} \mathcal{E}_{iu}$. Lemmas from 57 to 63 bound the probability of each $\mathcal{E}_{iu}$. Theorem 64 proves that $\Pr(\mathcal{E}_u)$ is $\mathcal{O}\left(\log^{-1} n\right)$. Corollary 65 states that $\mathcal{E}_u$ occurs with small probability. Theorem 66 proves that the expected number of three-cycles in G_n and not in G'_n is $\mathcal{O}\left(n^2 / \log n\right)$.

Lemma 57. $\Pr(\mathcal{E}_{1u}) < 3 \ln^{-1}(n+1)$.

Proof. Let $A = \{w \in V | d_{uw} = 2\}$. Note that $\mathcal{E}_{1u} = \bigcup_{a \in A} \mathcal{C}_{au}$. Using union bound, Definition 53, Lemma 54 and the facts that $d_{au} = 2$ for all $a \in A$ and $|A| \leq 6 \cdot 2 = 12$, then

$$\Pr(\mathcal{E}_{1u}) \leq \sum_{a \in A} \Pr(\mathcal{C}_{au}) \leq \sum_{a \in A} Z_a \cdot d_{au}^{-2} < 12 \cdot (\ln(n+1))^{-1} \cdot 2^{-2} = 3 \ln^{-1}(n+1).$$

□

Lemma 58. $\Pr(\mathcal{E}_{2u}) < 9/2 \ln^{-1}(n+1)$.

Proof. Let $A = \{w \in V | d_{uw} = 1\}$ and $B_a = \{w \in V | d_{aw} = 1\}$ for all $a \in A$. As OSW does not generate parallel edges when $(u, v) \in E'$ and $\mathcal{C}_{uv}$ occurs, then

$$\mathcal{E}_{2u} = \bigcup_{a \in A} \bigcup_{b \in A \setminus (\{a\} \cup B_a)} \mathcal{C}_{ab}.$$

Using union bound, Definition 53, Lemma 54 and the facts that $d_{ab} = 2$, $|A| \leq 6$ and $|A \setminus (\{a\} \cup B_a)| = |A| - 3 \leq 3$ for all a and $b \in A \setminus (\{a\} \cup B_a)$, then

$$\Pr(\mathcal{E}_{2u}) \leq \sum_{a \in A} \sum_{b \in A \setminus (\{a\} \cup B_a)} Z_a \cdot d_{ab}^{-2} < 6 \cdot 3 \cdot (\ln(n+1))^{-1} \cdot 2^{-2} = 9/2 \ln^{-1}(n+1).$$

□

Lemma 59. $\Pr(\mathcal{E}_{3u}) < 36\zeta(3) \ln^{-2}(n+1)$.

Proof. Let $A = \{w \in V | d_{uw} = 1\}$ and $B_a = \{w \in V | d_{aw} = 1\}$ for all $a \in A$. Note that $\mathcal{E}_{3u} = \bigcup_{a \in A} \bigcup_{b \in V \setminus (A \cup B_a)} (\mathcal{C}_{ab} \cap \mathcal{C}_{bu})$. Using union bound, $\Pr(\mathcal{E}_{3u}) \leq \sum_{a \in A} \sum_{b \in V \setminus (A \cup B_a)} \Pr(\mathcal{C}_{ab} \cap \mathcal{C}_{bu})$. As $\mathcal{C}_{ab}$ and $\mathcal{C}_{bu}$ are mutually independent events, so $\Pr(\mathcal{C}_{ab} \cap \mathcal{C}_{bu}) = \Pr(\mathcal{C}_{ab}) \cdot \Pr(\mathcal{C}_{bu})$ for all a and $b \in V \setminus (A \cup B_a)$. By this fact, using Definition 53 and grouping the terms with the same value of d_{bu},

$$\Pr(\mathcal{E}_{3u}) \leq \sum_{a \in A} \sum_{b \in V \setminus (A \cup B_a)} Z_a d_{ab}^{-2} \cdot Z_b d_{bu}^{-2} = \sum_{a \in A} \sum_{i=2}^{2n} \sum_{\substack{b \in V \setminus (A \cup B_a) \\ d_{bu} = i}} Z_a Z_b \cdot d_{ab}^{-2} d_{bu}^{-2}.$$

Using Lemma 54 and the facts that $d_{ab} \geq d_{bu} - 1$ and $|V \setminus (A \cup B_a) : d_{bu} = i| \leq 6i$ for all a, b and i,

$$\Pr(\mathcal{E}_{3u}) < \ln^{-2}(n+1) \sum_{a \in A} \sum_{i=2}^{2n} \sum_{\substack{b \in V \setminus (A \cup B_a) \\ d_{bu} = i}} (i-1)^{-2} \cdot i^{-2} \leq 6 \ln^{-2}(n+1) \sum_{a \in A} \sum_{i=2}^{2n} (i-1)^{-2} \cdot i^{-1}.$$

The proof follows because $i^{-1} < (i-1)^{-1}$ for $i \geq 2$, $\sum_{i=2}^{2n} (i-1)^{-3} < \sum_{i=1}^{\infty} i^{-3} = \zeta(3)$, by Fact 36, and $|A| \leq 6$. □

Lemma 60. $\Pr(\mathcal{E}_{4u}) < 3 \ln^{-1}(n+1)$.

Proof. Let $A = \{w \in V | d_{uw} = 2\}$. Note that $\mathcal{E}_{4u} = \bigcup_{a \in A} \mathcal{C}_{ua}$. The proof follows similarly to the Lemma 57 proof, using the fact that $d_{ua} = d_{au}$ for all $a \in A$. □

Lemma 61. $\Pr(\mathcal{E}_{5u}) < 36\zeta(3) \ln^{-2}(n+1)$.

Proof. Let $A = \{w \in V | d_{uw} = 1\}$ and $B_a = \{w \in V | d_{aw} = 1\}$ for all $a \in A$. Note that $\mathcal{E}_{5u} = \bigcup_{a \in V \setminus (\{u\} \cup A)} \bigcup_{b \in B_a \setminus A} (\mathcal{C}_{ua} \cap \mathcal{C}_{bu})$. In a similar way of the beginning of the Lemma 59 proof and grouping the terms with the same value of d_{ua},

$$\Pr(\mathcal{E}_{5u}) \leq \sum_{i=2}^{2n} \sum_{\substack{a \in V \setminus (\{u\} \cup A) \\ d_{ua} = i}} \sum_{b \in B_a \setminus A} Z_u Z_b \cdot d_{ua}^{-2} d_{bu}^{-2}.$$

The proof follows using Lemma 54 and the facts that $d_{bu} \geq d_{ua} - 1$, $|V \setminus (\{u\} \cup A) : d_{ua} = i| \leq 6i$ and $|B_a| \leq 6$ for all a, $b \in B_a$ and $2 \leq i \leq 2n$. □

Lemma 62. $\Pr(\mathcal{E}_{6u}) < 36\zeta(3)\ln^{-2}(n+1)$.

Proof. Let $A = \{w \in V | d_{uw} = 1\}$ and $B_a = \{w \in V | d_{aw} = 1\}$ for all $a \in A$. Note that $\mathcal{E}_{6u} = \bigcup_{a \in V \setminus (\{u\} \cup A)} \bigcup_{b \in A \setminus B_a} (\mathcal{C}_{ua} \cap \mathcal{C}_{ab})$. The proof follows similarly to the Lemma 61 proof. □

Lemma 63. $\Pr(\mathcal{E}_{7u}) < 36(3\zeta(3) + 1/8) \cdot \ln(2n) \cdot \ln^{-3}(n+1)$.

Proof. Let $A = \{w \in V | d_{uw} = 1\}$ and $B_a = \{w \in V | d_{aw} = 1\}$ for all $a \in A$. Note that $\mathcal{E}_{7u} = \bigcup_{a \in V \setminus (\{u\} \cup A)} \bigcup_{b \in V \setminus (\{u,a\} \cup A \cup B_a)} (\mathcal{C}_{ua} \cap \mathcal{C}_{ab} \cap \mathcal{C}_{bu})$. Using union bound, the fact that the events $\mathcal{C}_{ua}$, $\mathcal{C}_{ab}$ and $\mathcal{C}_{bu}$ are mutually independent, Definition 53 and Lemma 54,

$$\Pr(\mathcal{E}_{7u}) \leq \ln^{-3}(n+1) \sum_{a \in V \setminus (\{u\} \cup A)} d_{ua}^{-2} \sum_{b \in V \setminus (\{u,a\} \cup A \cup B_a)} d_{ab}^{-2} d_{bu}^{-2}.$$

The inequalities

$$\text{(i)} \quad \sum_{b \in V \setminus (\{u,a\} \cup A \cup B_a)} d_{ab}^{-2} d_{bu}^{-2} < 6(3\zeta(3) + 1/8) \text{ and}$$
$$\text{(ii)} \quad \sum_{a \in V \setminus (\{u\} \cup A)} d_{ua}^{-2} \leq 6\ln(2n) \text{ hold.}$$

For (i), the sum is split in three others: for $d_{ab} < d_{ua}$, $d_{ab} = d_{ua}$ and $d_{ab} > d_{ua}$. When $d_{ab} < d_{ua}$, the value of $d_{ua} - d_{ab}$ is positive and the triangle inequality property can be used such that $\sum_{b \in V \setminus (\{u,a\} \cup A \cup B_a)} d_{ab}^{-2} d_{bu}^{-2} \leq \sum_{b \in V \setminus (\{u,a\} \cup A \cup B_a)} d_{ab}^{-2} (d_{ua} - d_{ab})^{-2}$. Grouping the terms of the sum with the same value of d_{ab} and using the fact that $|V \setminus (\{u,a\} \cup A \cup B_a) : d_{ab} = i| \leq 6i$,

$$\sum_{b \in V \setminus (\{u,a\} \cup A \cup B_a)} d_{ab}^{-2} d_{bu}^{-2} \leq \sum_{i=2}^{d_{ua}-1} \sum_{\substack{b \in V \setminus (\{u,a\} \cup A \cup B_a) \\ d_{ab} = i}} i^{-2} (d_{ua} - i)^{-2} \leq 6 \sum_{i=2}^{d_{ua}-1} i^{-1} (d_{ua} - i)^{-2}.$$

Rearranging the terms,

$$\sum_{i=2}^{d_{ua}-1} i^{-1} (d_{ua} - i)^{-2} = \left(\sum_{i=2}^{\lceil \frac{d_{ua}-1}{2} \rceil} i^{-1} (d_{ua} - i)^{-2} + \sum_{i=1}^{\lfloor \frac{d_{ua}-1}{2} \rfloor} (d_{ua} - i)^{-1} i^{-2} \right),$$

where the first sum in the parentheses is 0 for $d_{ua} = 3$. In both sums, $d_{ua} - i \geq i$ and, when $d_{ab} < d_{ua}$,

$$\sum_{b \in V \setminus (\{u,a\} \cup A \cup B_a)} d_{ab}^{-2} d_{bu}^{-2} < 6 \left(\sum_{i=1}^{\lceil \frac{d_{ua}-1}{2} \rceil} i^{-3} + \sum_{i=1}^{\lfloor \frac{d_{ua}-1}{2} \rfloor} i^{-3} \right) < 6 \left(2 \sum_{i=1}^{\infty} i^{-3} \right) = 6 \cdot 2\zeta(3).$$

When $d_{ab} = d_{ua}$, as $d_{bu} \geq 2$, $|V \setminus (\{u,a\} \cup A \cup B_a) : d_{ab} = d_{ua}| \leq 6d_{ua}$ and $d_{ua} \geq 2$, then

$$\sum_{b \in V \setminus (\{u,a\} \cup A \cup B_a)} d_{ab}^{-2} d_{bu}^{-2} \leq (1/4) \sum_{b \in V \setminus (\{u,a\} \cup A \cup B_a)} d_{ab}^{-2} \leq (1/4) \cdot 6d_{ua} \cdot d_{ua}^{-2} \leq 6/8.$$

When $d_{ab} > d_{ua}$, using the triangle inequality property, grouping the terms of the sum with the same value of d_{ab} and using the fact that $|V \setminus (\{u, a\} \cup A \cup B_a) : d_{ab} = i| \leq 6i$,

$$\sum_{b \in V \setminus (\{u,a\} \cup A \cup B_a)} d_{ab}^{-2} d_{bu}^{-2} \leq 6 \sum_{i=d_{ua}+1}^{2n} i^{-1}(i - d_{ua})^{-2} = 6 \sum_{i=1}^{2n-d_{ua}} (d_{ua} + i)^{-1} i^{-2} < 6 \sum_{i=1}^{\infty} i^{-3}.$$

Therefore, $\sum_{b \in V \setminus (\{u,a\} \cup A \cup B_a)} d_{ab}^{-2} d_{bu}^{-2} < 6(3\zeta(3) + 1/8)$.

For (ii), grouping the terms of the sum with the same value of d_{ua} and using the facts that $|V \setminus (\{u\} \cup A) : d_{ua} = i| \leq 6i$ and $\sum_{i=1}^{m} i^{-1} \leq \ln(m) + 1$,

$$\sum_{a \in V \setminus (\{u\} \cup A)} d_{ua}^{-2} = \sum_{i=2}^{2n} \sum_{\substack{a \in V \setminus (\{u\} \cup A) \\ d_{ua} = i}} d_{ua}^{-2} \leq 6 \sum_{i=2}^{2n} i^{-1} \leq 6(\ln(2n) + 1 - 1) = 6 \ln(2n).$$

□

Theorem 64. $\Pr(\mathcal{E}_u)$ *is* $\mathcal{O}\left(\log^{-1} n\right)$.

Proof. Remembering that $\mathcal{E}_u = \bigcup_{i=1}^{7} \mathcal{E}_{iu}$ and, using union bound and Lemmas from 57 to 63,

$$\Pr(\mathcal{E}_u) < 21/2 \ln^{-1}(n+1) + 108\zeta(3) \ln^{-2}(n+1) + 36(3\zeta(3) + 1/8) \cdot \ln(2n) \cdot \ln^{-3}(n+1).$$

As $\ln(2n)/\ln(n+1)$ is $\mathcal{O}(1)$, then $\Pr(\mathcal{E}_u)$ is $\mathcal{O}\left(\log^{-1} n\right)$. □

Corollary 65. $\mathcal{E}_u$ *occurs with small probability.*

Theorem 66. *The expected number of three-cycles in* G_n *and not in* G'_n *is* $\mathcal{O}\left(n^2/\log n\right)$.

Proof. Let C_u be the sets of all three-cycles that contain u and have at least one long-range edge, for all $u \in V$. Let C_{iu} be the partitions of C_u with all three-cycles that have the same edge sequence defined in the event $\mathcal{E}_{iu}$, for all $1 \leq i \leq 7$ and u. Note that the cycles in C_{iu} are sequences of three vertices such that u is the first vertex in all. Let X_u be the random variables that count the number of three-cycles that contain u and have at least one long-range edge, for all u. Let X_c be the Bernoulli random variables that are one if the three-cycle c is in G_n or 0 otherwise, for all $c \in C_u$. Note that X_u and X_c are distinct random variables. Given the definitions,

$$X_u = \sum_{c \in C_u} X_c = \sum_{i=1}^{7} \sum_{c \in C_{iu}} X_c.$$

The value of $\mathrm{E}\left[\sum_{c \in C_{iu}} X_c\right]$ for all $1 \leq i \leq 7$ is computed in a similar way of the sequence of Lemmas from 57 to 63. For $i = 1$, the value of $\mathrm{E}\left[\sum_{c \in C_{1u}} X_c\right] = \sum_{c \in C_{1u}} \Pr(X_c = 1)$, by linearity of expectation and the fact that X_c are Bernoulli random variables. The cycles in C_{1u} are enumerated in a similar way of the Lemma 57 proof. Let $A = \{v \in V | d_{uv} = 2\}$ and $B_w = \{v \in V | d_{wv} = 1\}$ for $w \in V$, then

$$\mathrm{E}\left[\sum_{c \in C_{1u}} X_c\right] = \sum_{a \in A} \sum_{b \in (B_u \cap B_a)} \Pr(\mathcal{C}_{au}) \leq 2 \sum_{a \in A} \Pr(\mathcal{C}_{au}) < 6 \ln^{-1}(n+1),$$

because $|B_u \cap B_a| \leq 2$ and $\sum_{a \in A} \Pr(\mathcal{C}_{au}) < 3 \ln^{-1}(n+1)$, as Lemma 57 shows. In fact, obtaining the upper bounds of the next $\mathrm{E}\left[\sum_{c \in C_{iu}} X_c\right]$, for $2 \leq i \leq 7$, follows similarly to

Lemmas from 58 to 63, except for $i = 4$, which follows similarly to $i = 1$. Let X be the random variable that counts the number of three-cycles with at least one long-range edge. Therefore,

$$X = 1/3 \sum_{u \in V} X_u = 1/3 \sum_{u \in V} \sum_{i=1}^{7} \sum_{c \in C_{iu}} X_c,$$

because each c is considered three times in distinct u's. By linearity of expectation, $\mathrm{E}[X] = 1/3 \sum_{u \in V} \sum_{i=1}^{7} \mathrm{E}\left[\sum_{c \in C_{iu}} X_c\right]$ and, then $\mathrm{E}[X] < \lambda\left(n^2 / \log n\right)$ by Lemma 50. □

As the size of G'_n is $\Theta\left(n^2\right)$, by Lemmas 50 and 51, and the number of long-range edges in G_n are at most $|V|$, by Definition 53, then the size of G_n is $\Theta\left(n^2\right)$. So, Theorem 66 also shows that the expected number of three-cycles in G_n and not in G'_n is sublinear in the size of G_n. Using Euler characteristic and Lemmas 50 and 51, the number of three-cycles in G'_n is $8n^2$. This implies in the Corollary 67 statement. Note that a similar statement can be shown for the expected number of four-cycles in `UTSW` graphs.

Corollary 67. *The expected number of three-cycles in an* `OSW` *graph* G_n *is* $\Theta\left(n^2\right)$.

6 A Compact Routing Scheme

This chapter presents a labeling algorithm that takes as the input a UTSW graph and outputs a two-dimensional toroidal vertex labeling function, defined in Section 4.2. The algorithm is used to define a compact routing scheme for the UTSW graphs, which is the last result of the chapter. It uses many results presented in Chapter 4. Then, the events $\mathcal{E}_u$ and $\mathcal{E}_{iu}$ referred here are the same defined in Section 4.3. Besides that, the event $\mathcal{C}_{uv}$ defined in Section 6.1 is the same defined in Section 4.1. However, the notation $\mathcal{C}$ corresponds to a set from Section 6.2 to the end of the chapter. This chapter also uses the two-dimensional square torus T generated by the UTSW model and the labeling function ℓ referred in Section 4.2. All results presented in this chapter are published in the technical report of Viertel and Vignatti (2018a) and were submitted for publishing to the Theoretical Computer Science journal.

The design of the labeling algorithm is based on the spanning torus identification, as described below. Note that a breadth-first search can be used to label the vertices of a two-dimensional square torus. However, a UTSW graph has a two-dimensional square torus together with long-range edges. So, at first, the algorithm tries to remove the long-range edges. This procedure consists in executing searches rooted in each vertex aiming to remove incident long-range edges. The algorithm executes a search with depth four rooted in each vertex, and creates a list of four-cycles rooted in it. It identifies lattice patterns, formalized by Definition 71, through combinations of the four-cycles in the list. If the algorithm identifies lattice patterns with only local edges incident to the root vertex, then the remaining incident edges are long-range edges and it removes them. The resulting graph is an "almost" torus with possibly a few sparsely distributed long-range edges that cannot be identified by the algorithm.

After that, the algorithm chooses a random vertex of the resulting graph that can be the initial reference to label the remaining vertices. Finally, the algorithm labels almost all vertices executing a breadth-first search in the resulting graph rooted in that chosen vertex. The **labeling algorithm** (LA), defined in Algorithm 6, executes five important procedures:

1. **Four-cycles search** (C4S), defined in the Algorithm 1;
2. **Lattice pattern recognition algorithm** (LPRA), defined in the Algorithm 2;
3. **Long-range edges removing algorithm** (LRERA), defined in the Algorithm 3;
4. **Reference system labeling algorithm** (RSLA), defined in the Algorithm 4;
5. **Arbitrary cross labeling algorithm** (ACLA), defined in the Algorithm 5.

Sections 6.1 to 6.6 present the definition and analysis of each of these procedures.

6.1 Bounded Search

The **four-cycles search**, C4S for short, is an algorithm that takes as input a UTSW graph $G = (V, E)$ and a vertex $u \in V$, and outputs the set of all four-cycles in G, rooted in u, composed by four distinct vertices. It is a depth-first search with depth d limited to four, starting from $d = 1$. If a vertex v_d, at depth $d = 4$, is neighbor of u, then the path from root u to leaf v_d corresponds to a four-cycle composed by distinct vertices and, thus, the algorithm adds this cycle to the returning set $\mathcal{S}$.

Note that each cycle appears twice in $\mathcal{S}$, in both directions, due to G being an undirected graph. Then, the algorithm removes all duplicates in $\mathcal{S}$. The Algorithm 1 presents C4S, which executes two procedures: **recursive four-cycles search** (RC4S) and **duplication removing algorithm** (DRA). The notation $\mathcal{N}_G(w)$ corresponds to the set of neighbors of $w \in V$ in G. Lemma 68 shows that C4S outputs a set of four-cycles with size that tends to four.

Algorithm 1: Four-cycles search

Instance : A UTSW graph $G = (V, E)$ and a vertex $u \in V$
Return : The set of all four-cycles in G rooted in u and composed by four distinct vertices

```
1  Algorithm C4S (G, u)
2      return DRA (RC4S (G, (u, nil, nil, nil), 1) )
3  Algorithm RC4S (G, (v_1, v_2, v_3, v_4), d)
4      if d = 4 then
5          if v_1 ∈ N_G(v_4) then return {(v_1, v_2, v_3, v_4)};
6          return ∅
7      S ← ∅
8      for each w ∈ N_G(v_d) \ {v_k | 1 ≤ k ≤ d} do
9          v_{d+1} ← w
10         S ← S ∪ RC4S (G, (v_1, v_2, v_3, v_4), d + 1)
11     return S
12 Algorithm DRA (S)
13     R ← ∅
14     for each (v_1, v_2, v_3, v_4) ∈ S do
15         if (v_1, v_4, v_3, v_2) ∉ R then R ← R ∪ {(v_1, v_2, v_3, v_4)} ;
16     return R
```

Lemma 68. C4S *outputs a set of four-cycles with expected size of at most* $4 + \mathcal{O}\left(\log^{-1} n\right)$ *for all* $u \in V$.

Proof. Let C be the set of all possible distinct four-cycles in G with root vertex u, with at least one long-range edge and without considering directions. Let X_c be the random variables for all $c \in C$ that are one if C4S outputs a set that contains c, or zero otherwise. Let L be the random variable that counts the number of four-cycles in the set returned by C4S. As shown in Section 4.3, set of four-cycles with at least one long-range edge can be partitioned in nine subsets according to the sequence of edges between the vertices. Let C_i be the subsets of C partitioned as the events $\mathcal{E}_{iu}$ presented in Section 4.3, for all $1 \leq i \leq 9$. Let $\mathcal{C}_{uv}$ be the event of

vertex u choosing the vertex v to create a long-range edge in UTSW. As C4S finds at least the four distinct cycles that belongs to the two-dimensional square torus T generated by UTSW, so $L = 4 + \sum_{c \in C} X_c$. Then, by the linearity of expectation,

$$\mathrm{E}[L] = 4 + \mathrm{E}\left[\sum_{i=1}^{9} \sum_{c \in C_i} X_c\right] = 4 + \sum_{i=1}^{9} \sum_{c \in C_i} \Pr(X_c = 1).$$

The bound of $\sum_{c \in C_1} \Pr(X_c = 1)$ is computed in a similar way of Lemma 38 proof. Note that there are at most three distinct cycles $c \in C_1$ with the same c_4, where c_k is the vertex in the position k of the cycle c and $c_1 = u$. Let $A = \{a \in V | d_{ua} = 3\}$. Then $\sum_{c \in C_1} \Pr(X_c = 1) \leq \sum_{a \in A} 3 \Pr(\mathcal{C}_{ua} \cup \mathcal{C}_{au})$. As $|A| \leq 4d_{ua} = 12$, by Lemma 30, $d_{ua} = d_{au} = 3$ and using the union bound and Definition 29, then $\sum_{c \in C_1} \Pr(X_c = 1) \leq 3 \cdot \Pr(\mathcal{E}_{1u})$. Such proof strategy can be used to find upper bounds on $\sum_{c \in C_i} \Pr(X_c = 1)$, for all $1 \leq i \leq 9$, with the Lemmas 38 to 46. Then,

1. $\sum_{c \in C_1} \Pr(X_c = 1) \leq 3 \cdot \Pr(\mathcal{E}_{1u})$;
2. $\sum_{c \in C_2} \Pr(X_c = 1) \leq 2 \cdot \Pr(\mathcal{E}_{2u})$;
3. $\sum_{c \in C_3} \Pr(X_c = 1) \leq 2 \cdot \Pr(\mathcal{E}_{3u})$;
4. $\sum_{c \in C_4} \Pr(X_c = 1) \leq \Pr(\mathcal{E}_{4u})$;
5. $\sum_{c \in C_5} \Pr(X_c = 1) \leq \Pr(\mathcal{E}_{5u})$;
6. $\sum_{c \in C_6} \Pr(X_c = 1) \leq \Pr(\mathcal{E}_{6u})$;
7. $\sum_{c \in C_7} \Pr(X_c = 1) \leq 2 \cdot \Pr(\mathcal{E}_{7u})$;
8. $\sum_{c \in C_8} \Pr(X_c = 1) \leq \Pr(\mathcal{E}_{8u})$;
9. $\sum_{c \in C_9} \Pr(X_c = 1) \leq \Pr(\mathcal{E}_{9u})$.

The proof follows similarly to the proof of Theorem 47. Then,

$$\begin{aligned} \mathrm{E}[L] < \; 4 + \; & 146/9 \ln^{-1}(n/2) + \\ & (89/2\zeta(3) + 583/128) \ln^{-2}(n/2) + \\ & (24\zeta(3) + 8)(\ln n + 1) \ln^{-3}(n/2) + \\ & (3/4\zeta(3) + 1/4)(\ln n + 1)^2 \ln^{-4}(n/2), \text{ for all } u \in V. \end{aligned}$$

□

The expected size of the set returned by C4S, denoted $\mathrm{E}[L]$ and bounded at the end of the Lemma 68 proof, decreases as a reciprocal logarithmic function to the constant four whereas n increases linearly. So, $\mathrm{E}[L]$ is maximum when n is minimum. When $n = 3$, the expected size of the set returned by C4S is $\mathrm{E}[L] < 1745$, but better values are obtained by increasing n. For example, C4S outputs sets with approximately size of 7.48 for a UTSW graph with $n = 10$ in some experimental simulations. This value decreases for greater values of n, as foreseen by the theoretical bound of Lemma 68. It is approximately 5.42 for $n = 100$ and 5.24 for $n = 150$. Next, Lemma 70 uses the results of the Lemma 69 and shows that C4S executes in expected constant time.

Lemma 69. *Let $\delta_G(u)$ be the degree of each $u \in V$ in G, the expected degree of u is $\mathrm{E}[\delta_G(u)] \leq 6$.*

Proof. Let $\mathcal{N}_T(u)$ be the set of neighbors of u in the torus T. In UTSW, the vertex u chooses one vertex $v \in V \setminus \{u\}$ and any vertex $w \in V \setminus \{u\}$ can choose u to create a long-range edge. Let X_u be the random variable that counts the number of vertices that chose u to create a long-range

edge. Let X_{wu} be the random variable that is one if the vertex $w \in V \setminus \{u\}$ choose u to create a long-range edge, or zero otherwise. As $X_u = \sum_{w \in V \setminus \{u\}} X_{wu}$, by the linearity of expectation, $\mathrm{E}[X_u] = \sum_{w \in V \setminus \{u\}} \Pr(X_{wu} = 1)$. Using the Definitions 29 and 28 and $d_{uw} = d_{wu}$, then $\mathrm{E}[X_u] = 1$. Thus, using the linearity of expectation, $\mathrm{E}[\delta_G(u)] \leq |\mathcal{N}_T(u)| + 1 + \mathrm{E}[X_u] = 6$. □

Lemma 70. `C4S` *executes in expected constant time in any vertex* $u \in V$.

Proof. Let X_d be the random variable that counts the number of vertices at levels $1 \leq d \leq 4$ of `RC4S`. Let x_{d-1} be the already known number of vertices at level $d-1$. Let D_k be the random variable of the degree of each vertex $1 \leq k \leq x_{d-1}$ at level $d-1$. Note that $X_0 = x_0 = 1$ and $X_d = \sum_{k=1}^{x_{d-1}} D_k$ for all $1 \leq d \leq 4$. Then

$$\begin{aligned}
\mathrm{E}[X_d | X_{d-1} = x_{d-1}] &= \mathrm{E}\left[\sum_{k=1}^{x_{d-1}} D_k | X_{d-1} = x_{d-1}\right] \\
&= \sum_{k=1}^{x_{d-1}} \mathrm{E}[D_k | X_{d-1} = x_{d-1}] \\
&= \sum_{k=1}^{x_{d-1}} \mathrm{E}[D_k] \\
&\leq 6x_{d-1},
\end{aligned}$$

due to the linearity of expectation, to the independence between the variables D_k and X_{d-1} for all $1 \leq k \leq x_{d-1}$ and $1 \leq d \leq 4$, and because $\mathrm{E}[D_k] = \mathrm{E}[\delta_G(u)] \leq 6$ for all $u \in V$, by Lemma 69. So,

$$\begin{aligned}
\mathrm{E}[X_d] &= \sum_{x_{d-1} \geq 0} \mathrm{E}[X_d | X_{d-1} = x_{d-1}] \Pr(X_{d-1} = x_{d-1}) \\
&\leq 6 \sum_{x_{d-1} \geq 0} x_{d-1} \Pr(X_{d-1} = x_{d-1}) \\
&= 6\mathrm{E}[X_{d-1}].
\end{aligned}$$

Solving the recurrence with base $\mathrm{E}[X_0] = 1$, $\mathrm{E}[X_d] \leq 6^d$ for $d \geq 0$. Let X be the random variable that counts the total number of vertices at levels $0 \leq d \leq 4$ of the `RC4S`. As $X = \sum_{d=0}^{4} X_d$, then $\mathrm{E}[X] \leq \sum_{d=0}^{4} 6^d = 1555$. The result follows because `RC4S` visits at most an expected constant number of vertices, as shown in the bound of $\mathrm{E}[X]$, and because `DRA` takes as input a list of cycles with at most constant expected size, by Lemma 68. □

6.2 Lattice Pattern and Detection

Definition 71. Given a set $\mathcal{C}$ of four cycles of size four, the **lattice pattern** property consists in the existence of a sequence (C^0, C^1, C^2, C^3) of the four cycles $C^k \in \mathcal{C}$ where for all $0 \leq k \leq 3$:

1. the cycle C^k has a same vertex u;
2. the pair of consecutive cycles C^k and $C^{((k+1) \bmod 4)}$ have a distinct intersecting edge $\{u, a_k\}$;
3. the cycle C^k has a distinct vertex $b_k \neq u$ neighbor of $a_{((k-1) \bmod 4)}$ and a_k;

4. all nine vertices u, a_k and b_k are distinct each other.

The notation $\mathcal{C}$ corresponds to a set of four-cycles rather than an event from this section. A sequence of cycles with the properties of the Definition 71 may start with any cycle in $\mathcal{C}$. In fact, if $\mathcal{C}$ is a lattice pattern, then there are four distinct sequences with the lattice pattern properties. Note that if the sequence (C^0, C^1, C^2, C^3) is a lattice pattern, then are the sequences (C^1, C^2, C^3, C^0), (C^2, C^3, C^0, C^1) and (C^3, C^0, C^1, C^2) also. Figure 6.1 illustrates a lattice pattern with a sequence of cycles in the clockwise.

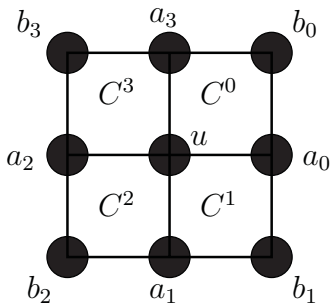


Figure 6.1: Four four-cycles that constitutes a lattice pattern.

The **lattice pattern recognition algorithm**, LPRA for short, takes as input a set $\mathcal{C}$ of four cycles of size four, with the same root vertex $u \in V$ and composed by distinct vertices, and outputs a boolean value that informs if the cycles define a lattice pattern. It recognizes a lattice pattern finding a sequence of the cycles in $\mathcal{C}$ that has the four properties of the Definition 71. The property (1) of the lattice pattern is assumed to be true by the input constraints.

LPRA, defined in Algorithm 2, starts selecting an arbitrary cycle $C^0 \in \mathcal{C}$, which is the first cycle of the sequence (C^0, C^1, C^2, C^3) without loss of generality. Note that $C^k = \left(c_1^k, c_2^k, c_3^k, c_4^k\right)$ is the cycle $0 \leq k \leq 3$ of the sequence and c_i^k is the vertex $1 \leq i \leq 4$ of the cycle k of the sequence. Moreover, $c_1^k = u$ for all k, by the input constraints, and either $a_0 = c_2^0$ or $a_0 = c_4^0$. Without loss of generality, the LPRA sets a_0 to c_4^0, removes C^0 from $\mathcal{C}$ and initializes the set S on lines 3 to 5. In this case, $a_0 = c_4^0$, $b_0 = c_3^0$ and $a_3 = c_2^0$ in the lattice pattern shown in Figure 6.1. LPRA uses the set S to check the property (4) in each iteration of the loop in the line 6.

After that, LPRA tries to find iteratively the next cycle of sequence C^k through the edge $\{u, a_{k-1}\}$, keeping the property (2) true. Then, either $a_{k-1} = c_2^k$ or $a_{k-1} = c_4^k$, because $c_1^k = u$ and considering both directions of the cycle. If there is no such $C^k \in \mathcal{C}$, then there is no sequence of the cycles in $\mathcal{C}$ with the properties of the Definition 71 and $\mathcal{C}$ does not define a lattice pattern.

The line 9 checks the property (4) in the cycles C^0 to C^k in each iteration $1 \leq k \leq 3$. When $k = 3$ and $\mathcal{C}$ defines a lattice pattern, either $c_2^3 = c_2^0$ or $c_4^3 = c_2^0$. This is the reason that LPRA does not add c_2^0 in S on line 4. However, this equality is checked on line 13, keeping the property (2) true. Lines from 10 to 12 removes C^k from $\mathcal{C}$ for LPRA selecting C^k only once; adds the vertices of C^k in S for checking the property (4) and sets a_k for finding the C^k of the next iteration. In the end, LPRA computes a sequence (C^0, C^1, C^2, C^3) and checks on line 13 the property (2) for the consecutive cycles C^3 and C^0.

LPRA does not consider the property (3). As all vertices of C^k are distinct each other, for all $C^k \in \mathcal{C}$, by the input constraints, so the property (3) is true for all $0 \leq k \leq 3$. Therefore, as the properties (1) and (3) are assumed to be true, the cycles C^k are found based on property (2), and the property (4) is checked in each iteration, then LPRA recognizes lattice patterns. Moreover, the running time of LPRA is constant, due to the set $\mathcal{C}$ having size four.

Algorithm 2: Lattice pattern recognition algorithm

Instance : A set $\mathcal{C}$ of four cycles of size four with the same root vertex and composed by distinct vertices

Return : A boolean value that informs if $\mathcal{C}$ defines a lattice pattern

Algorithm LPRA($\mathcal{C}$)
- Choose an arbitrary cycle $C^0 \in \mathcal{C}$, where $C^0 = (c_1^0, c_2^0, c_3^0, c_4^0)$
- $\mathcal{C} \leftarrow \mathcal{C} \setminus \{C^0\}$
- $S \leftarrow \{c_1^0, c_3^0, c_4^0\}$
- $a_0 \leftarrow c_4^0$
- **for** $k \leftarrow 1$ **to** 3 **do**
  - Find a cycle $C^k \in \mathcal{C}$, where $C^k = \left(c_1^k, c_2^k, c_3^k, c_4^k\right)$, such that $c_2^k = a_{k-1}$ **or** $c_4^k = a_{k-1}$
  - **if** such cycle C^k does not exist **then return** `false`;
  - **if** $\left\{c_2^k, c_3^k, c_4^k\right\} \setminus \{a_{k-1}\} \cap S \neq \emptyset$ **then return** `false`;
  - $\mathcal{C} \leftarrow \mathcal{C} \setminus \left\{C^k\right\}$
  - $S \leftarrow S \cup \left\{c_2^k, c_3^k, c_4^k\right\}$
  - $a_k \leftarrow \left\{c_2^k, c_4^k\right\} \setminus \{a_{k-1}\}$
- **if** $a_3 = c_2^0$ **then return** `true`;
- **return** `false`

6.3 Removing Long-Range Edges

Note that C4S outputs a set of four-cycles with the same root vertex and composed by distinct vertices. Also, this set has at least four cycles that define a lattice pattern, because a UTSW graph G has a spanning two-dimensional torus T. Then, it is possible to check the property of lattice pattern in all combinations of four cycles in the set that C4S outputs. The design of the next algorithm is based on this and its aim is to remove a large fraction of the long-range edges in G.

Let $\mathcal{L}$ be the set of four-cycles returned by C4S with input G and the root vertex $u \in V$. Let $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k \subseteq \mathcal{L}$ be all the $k \geq 1$ distinct combinations of four distinct cycles in $\mathcal{L}$ with lattice pattern. Let $U_i = (V_i, E_i)$ be the graph induced by the edges of the cycles in $\mathcal{C}_i$, for all $1 \leq i \leq k$. Let $U = \left(\bigcup_{i=1}^{k} V_i \ , \ \bigcup_{i=1}^{k} E_i\right)$. All vertices in the torus T have degree four, because T is a two-dimensional square torus. So, if $\delta_U(u) = 4$, then all edges incident to u in U are the local edges of u in T, remembering that $\delta_H(v)$ is the degree of v in the graph H. As a consequence, the set of all long-range edges of u in G is $\partial_G(u) \setminus \partial_U(u)$, where $\partial_H(v)$ is the set of all edges incident to v in the graph H. A vertex $u \in V$ is **detected** if $\delta_U(u) = 4$ and this is a boolean attribute of each vertex of G. Note that some vertices of G may not be detected.

The **long-range edges removing algorithm**, LRERA for short, takes as input a UTSW graph G and outputs a graph (V, E') without the long-range edges of all detected vertices. As explained, if a vertex $u \in V$ is detected, then all its incident edges in G can be identified as local or long-range edges. If the LRERA detects $u \in V$, then it does not add in E' all edges in $\partial_G(u) \setminus \partial_U(u)$, adding only the local edges of u in T. The algorithm does this through an attribute **add** of the edges of G.

The LRERA, defined in Algorithm 3, initializes the add attribute of each edge with the value `true`. After that, it tries to detect each $u \in V$. If a vertex u is detected, then it sets to

Algorithm 3: Long-range edges removing algorithm

Instance : A UTSW graph $G = (V, E)$
Return : A graph (V, E') without the detected long-range edges and the detected attribute defined for each $u \in V$

```
1  Algorithm LRERA(G)
2      for each e ∈ E do e.add ← true;
3      for each u ∈ V do
4          L ← C4S(G, u)
5          E'' ← ∅
6          for each C ∈ (L choose 4) do
7              if LPRA(C) then
8                  E'' ← E'' ∪ {edges in C that are incident to u}
9          if |E''| = 4 then
10             u.detected ← true
11             for each e ∈ ∂_G(u) \ E'' do e.add ← false;
12         else u.detected ← false;
13     E' ← ∅
14     for each e ∈ E do
15         if e.add then E' ← E' ∪ {e};
16     return (V, E')
```

false the add attribute of each long-range edge of u. At the end, LRERA adds in E' all edges in G where add is true. Furthermore, it sets to true the detected attribute of all detected vertices, otherwise sets to false for the remaining vertices. Later, the procedures **reference system labeling algorithm** (Algorithm 4) and **arbitrary cross labeling algorithm** (Algorithm 5) use such attribute.

LRERA executes C4S (Algorithm 1), that outputs the set $\mathcal{L}$ of all four-cycles in G with root vertex u. After that, it adds in E'' the incident edges of u for all combination $\mathcal{C}$ of four distinct cycles in $\mathcal{L}$ that LPRA recognizes as lattice pattern. That is, E'' is the set of incident edges of u in the graph U. So, $|E''| = \delta_U(u)$ and, as a consequence, if $|E''| = 4$ then all edges in E'' belong to the spanning torus T. Then, the algorithm sets to false the add attributes of each edge in $\partial_G(u) \setminus E''$. Lemmas 72, 73 and 74 show, respectively, that LRERA executes in expected linear time, LRERA detects a vertex with high probability and LRERA detects almost all vertices.

Lemma 72. LRERA *executes in expected linear time on* $|V|$.

Proof. C4S executes in expected constant time, by Lemma 70. The expected size of $\mathcal{L}$ is constant, due to Lemma 68. The running time of LPRA is constant. In line 11, note that, by Lemma 69, $\mathrm{E}[\delta_G(u)] \leq 6$, and $|E''| = 4$ by line 9. So line 11 executes in expected constant time because $\mathrm{E}[|\partial_G(u) \setminus E''|] \leq 2$. Then, the lines 4 to 12 execute in expected constant time. The other loops execute in linear time on $|V|$, because $|E| \leq 3|V|$ by the fact that UTSW generates the spanning torus $T = \left(V, \hat{E}\right)$ such that $\left|\hat{E}\right| = 2|V|$ and at most one long-range edge per vertex. Combining these, the result follows. □

Lemma 73. *Let* $\mathcal{D}_u$ *be the event of* LRERA *detecting* $u \in V$ *and* $\mathcal{E}_u$ *be the event of existence of at least one four-cycle, rooted in* u*, that is in* G *but not in* T*. Then,* $\Pr(\mathcal{D}_u) \geq 1 - 4\Pr(\mathcal{E}_u)$.

Proof. The lattice pattern is a well defined sequence of four cycles of size four. So, the probability of existence of at least one four-cycle in u with at least one long-range edge in any position of the sequence (C^0, C^1, C^2, C^3) bounds the probability of LRERA does not detect u. Let $\mathcal{A}_k$ be the event $\mathcal{E}_u$, as defined in Section 4.3, on the position k of the lattice pattern sequence, for all $0 \leq k \leq 3$. Then, the probability of LRERA does not detect u is $\Pr\left(\overline{\mathcal{D}_u}\right) \leq \Pr\left(\bigcup_{k=0}^{3} \mathcal{A}_k\right)$. Using the union bound, $\Pr\left(\overline{\mathcal{D}_u}\right) \leq 4\Pr(\mathcal{E}_u)$ and, then, $\Pr(\mathcal{D}_u) \geq 1 - 4\Pr(\mathcal{E}_u)$. □

Lemma 74. *The expected number of vertices that* LRERA *detects is* $|V|\left(1 - \mathcal{O}\left(\log^{-1} n\right)\right)$.

Proof. For each $u \in V$, let D_u be a random variable, such that $D_u = 1$ if LRERA detects u, $D_u = 0$ otherwise. Let D be the random variable that counts the number of vertices that LRERA detects. So, $D = \sum_{u \in V} D_u$ and, by the linearity of expectation, $\mathrm{E}[D] = \sum_{u \in V} \mathrm{E}[D_u] = \sum_{u \in V} \Pr(D_u = 1)$. The result follows by Lemma 73 and by the proof of the Theorem 47. Then,

$$\begin{aligned}\mathrm{E}[D] \geq |V|(1- \quad 4(\quad & 70/9\ln^{-1}(n/2) + \\ & (65/2\zeta(3) + 403/128)\ln^{-2}(n/2) + \\ & (24\zeta(3) + 8)(\ln n + 1)\ln^{-3}(n/2) + \\ & (3/4\zeta(3) + 1/4)(\ln n + 1)^2\ln^{-4}(n/2))).\end{aligned}$$

□

Lemma 74 means that the number of detected vertices increases as a reciprocal logarithmic function to $|V|$ whereas n increases linearly. The LRERA detection rate is about 88%, 96.04% and 96.35% of the vertices, respectively, for $n = 10, n = 100$ and $n = 150$ in some experimental simulations. Indeed, LRERA detects all vertices when n goes to infinity, as Corollary 75 shows.

Corollary 75. *The expected number of vertices that* LRERA *detects tends to* $|V|$ *as* n *goes to infinity.*

Proof. Let D be the random variable that counts the number of vertices that LRERA detects. Then,

$$\begin{aligned}\lim_{n\to\infty} \mathrm{E}[D] \geq |V| \quad (1 - 4 \lim_{n\to\infty} (\quad & 70/9\ln^{-1}(n/2) + \\ & (65/2\zeta(3) + 403/128)\ln^{-2}(n/2) + \\ & (24\zeta(3) + 8)(\ln n + 1)\ln^{-3}(n/2) + \\ & (3/4\zeta(3) + 1/4)(\ln n + 1)^2\ln^{-4}(n/2))) \\ = |V|.\end{aligned}$$

□

6.4 Labeling the Reference System

Remembering that the main purpose of the **labeling algorithm** (Algorithm 6) is to find a two-dimensional toroidal vertex labeling function for a spanning torus of the UTSW graph G, as explained in Section 4.2. In order to label the vertices and obtain a labeling ℓ_T, it is necessary a reference system. A **reference system** in G is (i) an **origin** that corresponds to a vertex $o \in V$ with label $(0, 0)$ and (ii) the four neighbors $o_1, o_2, o_3, o_4 \in V$ of o in T with the labels $(0, 1)$,

$(1,0)$, $(0,n-1)$ and $(n-1,0)$. The **cross rooted in** u is the vertex $u \in V$ and its four neighbors in T. The labeling algorithm initializes the reference system by finding a possible origin and labeling the cross rooted in it.

The LRERA (Algorithm 3) detects a large fraction of the vertices, thus, a large fraction of the long-range edges is removed from G. So, the returned graph $T' = (V, E')$ is an "almost" torus. Next, a vertex $o \in V$ is selected to become the origin if o and its neighbors in T' have all their long-range edges removed. I.e., the algorithm sets the vertex o as the origin if LRERA detected it and its neighbors. Note that, such o and its neighbors have degree equal to four in T'. Thus, it is possible to label the first cross and setting it as the reference system of G.

The **reference system labeling algorithm**, RSLA for short, takes as input the graph T' returned by LRERA and outputs a queue of vertices Q and a labeling vector ℓ, both already initialized. Its main purpose is to initialize a breadth-first search that labels the remaining vertices (whenever possible), starting from the labeled vertices of the reference system. It finds a vertex o that can be the origin, labels the cross rooted in o generating the reference system and enqueues in Q only the neighbors of o. The queue Q is used later to perform the breadth-first search.

Algorithm 4: Reference system labeling algorithm

Instance : A graph ("almost" torus) $T' = (V, E')$
Return : A queue Q and the labeling vector ℓ with the reference system vertices labeled

```
1  Algorithm RSLA(T')
2      for each u ∈ V do
3          ℓ[u] ← nil
4          u.enqueued ← false
5      repeat
6          Choose o ∈ V independently at random
7      until o and its neighbors in T' are all detected;
8      ℓ[o] ← (0, 0)
9      (o_1, o_2, o_3, o_4) ← RSVFA(T', o)
10     ℓ[o_1] ← (0, 1), ℓ[o_2] ← (1, 0), ℓ[o_3] ← (0, n − 1), ℓ[o_4] ← (n − 1, 0)
11     Q ← ∅
12     o.enqueued ← true
13     for i ← 1 to 4 do
14         Q ← Q ∪ {o_i}
15         o_i.enqueued ← true
16     return (Q, ℓ)
17 Algorithm RSVFA(T', o)
18     C ← C4S(T', o)
19     Choose an arbitrary cycle (c_1, c_2, c_3, c_4) ∈ C, where c_1 = o
20     o_1 ← c_2, o_2 ← c_4
21     for i ← 3 to 4 do
22         Find (c_1, c_2, c_3, c_4) ∈ C, where c_1 = o, such that
             (c_2 = o_{i−1} ∧ c_4 ≠ o_{i−2}) ∨ (c_4 = o_{i−1} ∧ c_2 ≠ o_{i−2})
23         o_i ← {c_2, c_4} \ {o_{i−1}}
24     return (o_1, o_2, o_3, o_4)
```

RSLA, defined in Algorithm 4, starts assigning the labels of all vertices to nil value and their **enqueued** attribute to false. After, it chooses $o \in V$ that can be a potential origin for the reference system, on lines 5 to 7. If o and its neighbors in T' are detected, then RSLA sets o as the origin. It labels o as $(0, 0)$ and finds the neighbors of o in T' calling the **reference system vertices finder algorithm**, RSVFA for short. RSVFA executes C4S and adds in $\mathcal{C}$ all four-cycles in T' with root vertex in o. As o is detected, so $\delta_{T'}(o) = 4$, and each neighbor of o in the cycles in $\mathcal{C}$ is one of the four vertices (ii) of the reference system.

When the size of the torus is $n = 4$, $\mathcal{C}$ has two four-cycles rooted in o composed only by local edges that wrap around the two-dimensional square torus T. This causes the labeling algorithm presented in this thesis to fail. Thus, it is considered tori of sizes $n \geq 5$. As $n \geq 5$ and o and its neighbors have degree four in T', so there are only four distinct cycles in $\mathcal{C}$ and these define a lattice pattern. So, RSVFA chooses an arbitrary cycle $(c_1, c_2, c_3, c_4) \in \mathcal{C}$ and sets the first two vertices of the reference system o_1 and o_2 to c_2 and c_4, respectively. This is possible because C4S outputs cycles so that $c_1 = o$, $c_2 \neq c_4$ and because $\mathcal{C}$ defines a lattice pattern. RSVFA finds o_3 and o_4 in the cycles in $\mathcal{C}$ in a similar way that LPRA does.

After RSVFA finding the vertices of the reference system $o_1, \ldots, o_4$, RSLA labels them on line 10. Note that the algorithm computes n as $|V|^{1/2}$. Finally, it enqueues the neighbors of o in Q and tags the first cross as already enqueued, in order to does not enqueue these vertices again in the breadth-first search, performed by the **labeling algorithm** (LA), described in Section 6.6. Note that it is not necessary to enqueue o because o and its neighbors are already labeled. Lemma 76 bounds the expected running time of lines from 5 to 7.

Lemma 76. *Lines 5 to 7 of* RSLA *execute in expected constant time, when receiving an output of* LRERA.

Proof. Let X be the random variable that counts the number of vertices that RSLA chooses until it finds an origin for the reference system. Note that X is a geometric random variable. Let $\mathcal{D}_w$ be the event of LRERA detecting the vertex $w \in V$. The probability of a $o \in V$ being an origin is $p = \Pr\left(\mathcal{D}_o \cap \bigcap_{v \in \mathcal{N}_T(o)} \mathcal{D}_v\right)$, where $\mathcal{N}_T(o)$ is the set of neighbors of o in the original spanning torus T of G generated by the UTSW model. By De Morgan's laws, $p = 1 - \Pr\left(\overline{\mathcal{D}_o} \cup \bigcup_{v \in \mathcal{N}_T(o)} \overline{\mathcal{D}_v}\right)$. By Lemma 73, $\Pr\left(\overline{\mathcal{D}_w}\right) \leq 4\Pr(\mathcal{E}_w)$ for all $w \in V$. The result follows by the union bound, Theorem 47 proof and by the fact that X is a geometric random variable. Then,

$$\begin{aligned}\mathrm{E}[X] < \quad (1 - (\quad & 1400/9 \ln^{-1}(n/2) + \\ & (650\zeta(3) + 2015/32) \ln^{-2}(n/2) + \\ & (480\zeta(3) + 160)(\ln n + 1) \ln^{-3}(n/2) + \\ & (15\zeta(3) + 5)(\ln n + 1)^2 \ln^{-4}(n/2)))^{-1}.\end{aligned}$$

□

The running time of Algorithm 4 depends on: (i) the running time of the initializations of the labels and enqueued attribute of the vertices on lines 2 to 4; (ii) the running time of the lines 5 to 7 and (iii) the running time of RSVFA, which depends on the running time of C4S on line 18. Lemmas 76 and 70 bound (ii) and (iii), respectively, and (i) is linear on $|V|$. Combining these, Corollary 77 follows.

Corollary 77. RSLA *executes in expected linear time on* $|V|$*, when receiving an output of* LRERA.

6.5 Labeling Arbitrary Crosses

The previous section shows how to label the vertices of the reference system. After that procedure, it is necessary to label the remaining vertices of the graph $T' = (V, E')$ that LRERA outputs. It is possible to label the cross rooted in a detected and labeled vertex $u \in V$ if there is a lattice pattern rooted in u with some properties. The algorithm of this section performs this labeling.

The **arbitrary cross labeling algorithm**, ACLA for short, takes as input the graph T', a detected and labeled vertex u, a queue of vertices Q and a labeling vector ℓ. It outputs the queue Q with the cross rooted in u enqueued and the labeling vector ℓ with the same cross labeled. The algorithm starts executing C4S to find the set $\mathcal{L}$ of all four-cycles rooted in u. The set $\mathcal{L}$ has at least one lattice pattern $\mathcal{C} \subseteq \mathcal{L}$ composed by a cycle with detected and labeled vertices such that these vertices can be used to label the cross rooted in u. Such claim holds because the **labeling algorithm** (LA), defined in Algorithm 6, executes a breadth-first search, that already labeled some vertices of at least one cycle in $\mathcal{C}$ that can be used in the labeling. Section 6.6 shows this with more details in Theorem 79.

Based on that claim, there is a cycle (c_1, c_2, c_3, c_4) in a lattice pattern $\mathcal{C} \subseteq \mathcal{L}$ that provides a reference. A **reference** (do not confuse with **reference system**) is a pair of local edges in (c_1, c_2, c_3, c_4) with labeled endpoints and that are perpendicular in the lattice pattern $\mathcal{C}$. Two local edges are **perpendicular** in $\mathcal{C}$ if they are consecutive in (c_1, c_2, c_3, c_4). The Figure 6.2 illustrates these definitions, where $c_1 = u$. The bold edges represent perpendicular edges. So, the algorithm finds a cycle (c_1, c_2, c_3, c_4) with a reference and places the neighbors of u in the lattice pattern $\mathcal{C}$ on the four endpoints of the cross $u_1, \ldots, u_4$, in a similar way that RSLA does. After that, it labels $u_1, \ldots, u_4$ using the topological information in the reference of (c_1, c_2, c_3, c_4).

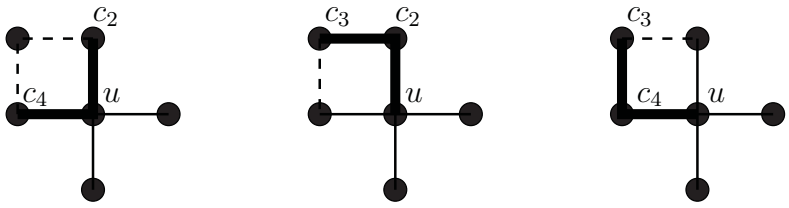


Figure 6.2: Possible references to label the cross.

Let $(c_1, c_2, c_3, c_4) \in \mathcal{C}$ be the four-cycle that has a reference for labeling the cross rooted in u. Remembering that $c_1 = u$ because C4S outputs $\mathcal{L}$. If c_2 and c_4 are labeled, then the edges $\{u, c_2\}$ and $\{u, c_4\}$ are a reference. These edges are perpendicular in $\mathcal{C}$ and are local edges because u is detected. The Figure 6.2 illustrates this case on the left side, where the bold edges are the reference. Besides that, (c_1, c_2, c_3, c_4) may have three more distinct possible references. If c_2 or c_3 are detected, then the edges $\{u, c_2\}$ and $\{c_2, c_3\}$ are local edges because u is detected. Moreover, if c_2 and c_3 are labeled, then these edges are a reference, by the fact that u is also labeled. Figure 6.2 illustrates this case in the middle. The same happens with vertices c_3 and c_4 such that the edges $\{u, c_4\}$ and $\{c_3, c_4\}$ are a reference. The Figure 6.2 illustrates this case in the right side. It is not necessary to consider the case where the edges $\{c_2, c_3\}$ and $\{c_3, c_4\}$ are a reference. The reason is that the last two cases occur simultaneously when this occurs, because u is detected and labeled.

The Algorithm 5 defines ACLA. The lines 2 to 15 find a lattice pattern $\mathcal{C}$ and a reference for labeling the cross rooted in u as explained above. It sets the first two vertices of the cross, u_1 and u_2, and labels one of them, if required. The lines 16 to 18 set the other two vertices, u_3 and u_4, in a similar way that RSVFA does. The ACLA finishes labeling the vertices of the cross,

enqueuing in Q those that are detected and are not enqueue yet, and assigning their enqueued attribute to `true` in order to not enqueue those vertices again.

Algorithm 5: Arbitrary cross labeling algorithm

Instance : A graph ("almost" torus) $T' = (V, E')$, a vertex $u \in V$, a queue Q and a labeling vector ℓ

Return : A queue Q and the labeling vector ℓ with the cross rooted in u labeled

```
1  Algorithm ACLA(T', u, Q, ℓ)
2      L ← C4S(T', u)
3      for each lattice pattern C ⊆ L do
4          for each (c1, c2, c3, c4) ∈ C, where c1 = u do
5              if ℓ[c2] ≠ nil ∧ ℓ[c4] ≠ nil then
6                  u1 ← c2, u2 ← c4
7                  Break both for
8              else if (c2.detected ∨ c3.detected) ∧ ℓ[c2] ≠ nil ∧ ℓ[c3] ≠ nil then
9                  u1 ← c2, u2 ← c4
10                 ℓ[u2] ← ℓ[u] + ℓ[c3] − ℓ[c2]
11                 Break both for
12             else if (c3.detected ∨ c4.detected) ∧ ℓ[c3] ≠ nil ∧ ℓ[c4] ≠ nil then
13                 u1 ← c2, u2 ← c4
14                 ℓ[u1] ← ℓ[u] + ℓ[c3] − ℓ[c4]
15                 Break both for
16     for i ← 3 to 4 do
17         Find (c1, c2, c3, c4) ∈ C, where c1 = u, such that
             (c2 = u_{i−1} ∧ c4 ≠ u_{i−2}) ∨ (c4 = u_{i−1} ∧ c2 ≠ u_{i−2})
18         u_i ← {c2, c4} \ {u_{i−1}}
19     ℓ[u3] ← 2ℓ[u] − ℓ[u1]
20     ℓ[u4] ← 2ℓ[u] − ℓ[u2]
21     for i ← 1 to 4 do
22         if (not u_i.enqueued) ∧ u_i.detected then
23             Q ← Q ∪ {u_i}
24             u_i.enqueued ← true
25     return (Q, ℓ)
```

The assignments of labels to the vertices in the lines 10, 14, 19 and 20 perform operations of two-dimensional vector addition and subtraction among the labels of the vertices. These operations may not result in elements of $[\![n]\!]^2$. The algorithm solves this by replacing each coordinate of the label $\ell[v]_j$ by $\ell[v]_j \bmod n$, where j is the label dimension of the vertex v. Combining Lemmas 70 and 68, together with the fact that `LPRA` executes in constant time, Corollary 78 follows.

Corollary 78. `ACLA` *executes in expected constant time, when receiving an output of* `LRERA`.

6.6 Main Algorithm

This section presents the main procedure of the labeling algorithm. It removes most of the long-range edges and executes a breadth-first search. The breadth-first search labels the crosses rooted in a large fraction of the vertices. The algorithm labels almost all vertices, executes in expected linear time and can be used to define a compact routing scheme for UTSW graphs.

The **labeling algorithm**, LA for short, takes as input a UTSW graph $G = (V, E)$ and outputs a vector ℓ that represents the labeling function $\ell_T : V \to [\![n]\!]^2$, where T is the original spanning torus of G generated by UTSW. It starts executing LRERA with G as input. The output is the torus T' with a few remaining long-range edges and with the detected attribute of all vertices already assigned. After that, the algorithm initializes a breadth-first search, finding a vertex that can be the origin, and labels the vertices of the reference system calling RSLA with T' as input. The algorithm executes the breadth-first search iteratively dequeing u from the queue Q and calling ACLA to label the cross rooted in u. ACLA iteratively labels the vertices of the crosses and enqueues in Q these vertices that were not enqueued yet. Algorithm 6 defines LA.

Algorithm 6: Labeling algorithm

Instance : A UTSW graph $G = (V, E)$
Return : A vector ℓ representing the labeling $\ell_T : V \to [\![n]\!]^2$

```
1 Algorithm LA(G)
2     T' ← LRERA(G)
3     (Q, ℓ) ← RSLA(T')
4     while Q ≠ ∅ do
5         Dequeue u from Q
6         (Q, ℓ) ← ACLA(T', u, Q, ℓ)
7     return ℓ
```

LA may not label some vertices of G, then ℓ defines a **partial** two-dimensional toroidal vertex labeling function. However, it labels most of them, because LRERA detects most of the vertices, as Lemma 74 shows. Besides that, some of the non-detected vertices are labeled in the end of LA execution. This occurs because if one of the four neighbors $w \in \mathcal{N}_T(v)$ of a non-detected vertex $v \in V$ in the original spanning torus T was enqueued in Q, then ACLA labels v, that are in the cross rooted in w.

Theorem 79 shows that LA labels the crosses rooted in all enqueued vertices. The breadth-first search executes only over the local edges, in consequence of visiting only detected vertices. Then, LA does not label a vertex $v \in V$ only if v is not reachable from the origin of the reference system $o \in V$ by a path composed only by local edges and detected vertices. That is, v is inside an area of T in which the boundaries are composed by non-detected vertices only.

Theorem 79. LA *labels all the vertices of crosses rooted in each vertex of* Q.

Proof. Let u_i be the vertex that is in the front of Q in the i^{th} iteration of line 4 in Algorithm 6. Let s_i be the root of the cross processed by RSLA or ACLA when u_i was enqueued in Q. Note that s_i and u_i are both detected, because RSLA chooses a detected origin and enqueues detected vertices, and ACLA enqueues only detected vertices. The edge $\{u_i, s_i\}$ belongs to the original spanning torus T, because both endpoints are detected. Let $c_{i1} = (u_i, s_i, v_1, w_1)$ and $c_{i2} = (u_i, s_i, v_2, w_2)$ be the two distinct four-cycles induced in T. The vertices u_i, v_1 and v_2 are all labeled, because

either RSLA already labeled them if s_i is the **origin**, or ACLA already labeled them during the labeling of the cross in s_i. Also, the vertex s_i is also labeled, because either RSLA already labeled s_i if it is the **origin**, or RSLA or ACLA already labeled it before enqueuing it in Q. Without loss of generality, $\{u_i, s_i\}$ and $\{s_i, v_1\}$ have labeled endpoints and are local edges perpendicular in the lattice pattern rooted in u_i and induced in T, i.e., they are a reference in c_{i1}. Let $\mathcal{L}_{u_i}$ be the list assigned on line 2 of ACLA when $u = u_i$. Note that c_{i1} are in $\mathcal{L}_{u_i}$. As c_{i1} has the two edges $\{u_i, s_i\}$ and $\{s_i, v_1\}$ as a reference and s_i is detected, so ACLA labels the vertices of the cross in u_i. Therefore, LA labels the cross of each vertex u_i enqueued in Q. □

Besides some non-labeled vertices may exist, this number is small. Corollary 75 shows that the number of detected vertices tends to $|V|$ whereas the size of the torus n grows. Equivalently, the number of non-detected vertices tends to zero. The combination of this fact with Theorem 79 results in Theorem 80 statement.

Theorem 80. *As n goes to infinity, the number of vertices that* LA *labels tends to $|V|$.*

Algorithm 6 executes LRERA and RSLA once, demanding expected linear time, by Lemma 72 and Corollary 77. Each iteration of the breadth-first search executes ACLA once, demanding expected constant time, by Corollary 78. As the breadth-first search executes at most $|V| - 1$ iterations, then the LA executes in expected linear time, as showed in Theorem 81.

Theorem 81. LA *executes in expected linear time on $|V|$.*

The main application of LA is in routing of messages in UTSW graphs. Section 4.2 claims that Kleinberg's greedy routing algorithm can route messages if a two-dimensional toroidal vertex labeling function ℓ (Definition 34) is known. In this sense, each vertex $u \in V$ requires $2\lceil \log n \rceil$ bits for its label $\ell[u]$, $\lceil \log n \rceil$ bits for each dimension, where n is the size of the torus. Also, u requires expected $\mathcal{O}(\log n)$ bits for its routing table, because each neighbor $v \in \mathcal{N}_G(u)$ of u in G has a row in u's routing table, encoded by the pair $(\ell[v], p(u, v))$, where $p(u, v)$ is the port number in u that directs the message to the edge in u incident to v in G. As $\ell[v]$ requires $2\lceil \log n \rceil$ bits, $p(u, v)$ requires expected $\mathcal{O}(1)$ bits and u's routing table has expected $\mathcal{O}(1)$ rows, both by Lemma 69, so u's routing table has expected $\mathcal{O}(\log n)$ bits. Then, each $u \in V$ requires expected $\mathcal{O}(\log n)$ bits to execute the Kleinberg's greedy routing algorithm, which is sublinear in the size of the network G, given $|V| = n^2$ and $|E| \leq 3|V|$.

Considering the case of generating the routing tables with a partial two-dimensional toroidal vertex labeling function, there is a fixed-port name-dependent compact routing scheme for G. The preprocessing algorithm generates sublinear structures per vertex and the routing algorithm forwards messages in expected constant time. Theorem 82 shows this.

Theorem 82. *There is a compact routing scheme for* UTSW *graphs with high probability.*

Proof. The following preprocessing algorithm generates the labels and routing tables for each vertex of G. Execute LA with input G, generating the labeling ℓ. For each $u \in V$, assign $\ell[u]$ to its label. For each $\{u, v\} \in E$, add $(\ell[v], p(u, v))$ in u's routing table and $(\ell[u], p(v, u))$ in v's routing table. LA generates the labeling ℓ in expected linear time, as Theorem 81 states. The preprocessing algorithm generates the labels and the routing tables in time $\Theta(|V|)$, because each access in ℓ and in p executes in constant time and because $2|V| \leq |E| \leq 3|V|$. So, the preprocessing algorithm executes in $\mathcal{O}(|V|)$ expected time and generates structures with expected $\mathcal{O}(\log n)$ bits for each vertex, as explained above. Kleinberg's greedy routing algorithm executes in expected constant time, by Lemma 69. Therefore, there is an expected linear time preprocessing algorithm that generates structures with expected sublinear size for each vertex and a related routing algorithm that executes in expected constant time on each vertex. □

7 Conclusion

This work presents two small world generative models. Both exhibit the two main properties of the small world networks: clustering and paths with small sizes. The first one is the undirected toroidal small world model (UTSW), defined in Section 4.1. It builds undirected graphs over a two-dimensional torus together with random long-range edges. The random edges generation may create cycles of size four outside the torus. However, UTSW generates these cycles rooted in a vertex with small probability. Such topological property inspired the design of a labeling algorithm, presented in Chapter 6. The routing of the messages in UTSW graphs is done by the Kleinberg's greedy routing algorithm, explained in the beginning of Section 3.1. It routes messages in a UTSW graph (V, E) with expected $\mathcal{O}\left(\log^2 n\right)$ number of forwardings, where $|V| = n^2$ and $|E| \leq 3n^2$.

The second small world model presented in this work is the octahedral small world (OSW), defined in Section 5.2. The OSW model generates directed graphs over an octahedron positioned at the center of the three-dimensional integer space $\mathbb{Z}^3$. It generates the base graph on the surface of an octahedron, distributing the vertices on the surface and connecting only the closest vertices from each other. After that, the model generates random long-range edges to model small paths. Section 5.1.1 relates the octahedron with spheres as explained below. All distances between the neighbors vertices on the surface of the octahedron is equal to $\sqrt{2}$ in the euclidean norm. There are spheres, for a given octahedron, such that all distances between the neighbors in the base graph projected on the surface of the sphere are bounded by a constant value. Then, there are a family of spheres that are asymptotically equivalent to the octahedron in the base graph generation. The OSW model is related to, for example, friendship networks over the Earth globe, where people are spread on the surface of a sphere and have close and far away friends. Section 5.2 presents a greedy routing algorithm that routes messages in an OSW graph (V, E) with expected $\mathcal{O}\left(\log^2 n\right)$ number of forwardings, where $|V| = 4n^2 + 2$ and $|E| \leq 25n^2$.

Section 4.2 defines the toroidal small world labeling problem. Kleinberg's greedy routing algorithm requires the knowledge of the positions in the lattice of all vertices for routing messages in UTSW graphs. Labeling the vertices with the positioning information of graphs generated by random small world models when the positions are unknown poses serious challenges. The structure of the base two-dimensional torus generated by UTSW model can be seen as a well-formed pattern of four-cycles. The difficulty in the vertices labeling arises from the random edges generation, which may create new four-cycles that do not belong to the torus. Indeed, in extreme cases, as Figure 4.2 shows, there may be more than one induced torus. However, UTSW generates four-cycles rooted in a vertex and outside the torus with small probability, which motivates the approach used in the design of the labeling algorithm presented Chapter 6. The labeling algorithm labels the vertices of a UTSW graph identifying the four-cycles rooted in each vertex that are in the two-dimensional torus. For large graphs, it assigns a position in the lattice for almost all vertices, as Theorem 80 states. Then, it can be used to define a compact routing scheme for UTSW graphs.

Chapter 6 defines a compact routing scheme specialized in UTSW graphs. It presents a preprocessing algorithm in Theorem 82 that uses the labeling algorithm as a subroutine. The preprocessing algorithm executes the labeling algorithm, which outputs a two-dimensional toroidal vertex labeling function, defined in Section 4.2. The labeling function is used to generate the labels and routing tables of all vertices, which are used by the Kleinberg's greedy routing algorithm. The preprocessing algorithm with a UTSW graph (V, E) as the input executes in expected $\mathcal{O}(|V|)$ time and generates labels and routing tables with expected $\mathcal{O}(\log n)$ bits, where $|V| = n^2$ and $|E| \leq 3n^2$. Then, the routing scheme has a preprocessing algorithm that generates compact data structures for all vertices and has a routing algorithm that routes messages through small paths. The routing scheme is fixed-port, that is, the network designer is free to set the vertices ids and ports numbers, where the former represents a unique number that identify the vertex and the latter identifies the edges of each vertex. Moreover, the routing scheme is name-dependent, that is, there are labels with network topological information assigned to the vertices.

In fact, the presented preprocessing algorithm may not generate the labels and routing tables of all vertices. The labeling algorithm finds a partial two-dimensional toroidal vertex labeling function, that is, there may be some vertices that do not have a label. These vertices are sparsely located in the two-dimensional torus when the graph is large and are in low number, that is, it tends to zero as the graph size goes to infinity. Cases showed in Figure 4.2 are rare to occur. As these vertices are few, so their positions can be assigned manually if their presence in the network are really necessary. Another extremely cases are "closed areas" of vertices on the two-dimensional torus bounded only by non-detected vertices, these defined in Section 6.3. The set of all vertices inside of an area of this type is a vertex cut in the generated graph. In this sense, let $A \subset V$ be the set of all vertices of the UTSW graph (V, E) that are bounded by non-detected vertices on the two-dimensional torus. If the labeling algorithm selects the origin $o \in V$, more precisely in Algorithm 4, and $o \in A$, then the labeling algorithm may label only the vertices in A. Otherwise, the labeling algorithm may label only the vertices in $V \setminus A$. This occur because the labeling algorithm labels the vertices performing a breadth-first search that visits only detected vertices. That is, the labeling algorithm does not reach all vertices in $V \setminus A$ if $o \in A$ and, as a consequence, the vertices in $V \setminus A$ are not labeled. However, as the number of non-detected vertices tends to zero as the graph size goes to infinity, so the sizes of all sets of vertices bounded by non-detected vertices also tends to zero. Then, even when this exceptional case occur, only very small areas of vertices remains unlabeled. Therefore, in most cases, the labeling algorithm performs well.

7.1 Future Works

For future works, Section 5.3 indicates a possibility of designing a labeling algorithm for OSW graphs. Corollary 67 shows that it is possible to perform a search of three-cycles in all vertices in expected linear time in the size of the network. Moreover, Corollary 65 presents an evidence that finding the long-range edge of each vertex may be possible. Then, one future work is the design of an algorithm that removes the long-range edges of OSW graphs and highlights the octahedral base graph. A similar approach can be used to design a labeling algorithm for other small world model in the literature, such those presented in Section 3.1. Other future work can consider graphs that better models the real world. For example, the torus and the octahedral graph used in this work model a situation where the vertices are equally distributed on a geographical space. Some models (Liben-Nowell et al., 2005; Kleinberg, 2001) deal with this issue, but it remains an open problem if the results of this work can be adapted to them. Also, several real-world

network models can be considered, for example, models for power-law graphs and the hyperbolic geometric graphs.

REFERENCES

Abraham, I., Gavoille, C., Goldberg, A. V., and Malkhi, D. (2006a). Routing in Networks with Low Doubling Dimension. In *26th IEEE International Conference on Distributed Computing Systems*, page 75, Lisbon - Portugal.

Abraham, I., Gavoille, C., and Malkhi, D. (2006b). On Space-Stretch Trade-Offs: Lower Bounds. In *Eighteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 207–216, Cambridge, Massachusetts - United States of America.

Abraham, I., Gavoille, C., Malkhi, D., Nisan, N., and Thorup, M. (2004a). Compact Name-Independent Routing with Minimum Stretch. In *Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 20–24, Barcelona, Catalonia - Spain.

Abraham, I., Malkhi, D., and Dobzinski, O. (2004b). LAND: Stretch ($1 + \varepsilon$) Locality-Aware Networks for DHTs. In *Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 550–559, New Orleans, Louisiana - United States of America.

Adamic, L. and Adar, E. (2005). How to search a social network. *Social Networks*, 27(3):187–203.

Arias, M., Cowen, L. J., Laing, K. A., Rajaraman, R., and Taka, O. (2003). Compact Routing With Name Independence. In *Fifteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 184–192, San Diego, California - United States of America.

Aspnes, J., Diamadi, Z., and Shah, G. (2002). Fault-tolerant Routing in Peer-to-peer Systems. In *Twenty-first Annual Symposium on Principles of Distributed Computing*, pages 223–232, Monterey, California - United States of America.

Awerbuch, B. and Peleg, D. (1990). Sparse Partitions. In *31st Annual Symposium on Foundations of Computer Science*, pages 503–513, St. Louis, Missouri - United States of America.

Bakun, O. and Konjevod, G. (2010). Adaptive decentralized routing in small-world networks. In *29th IEEE Conference on Computer Communications*, pages 1–6, San Diego, California - United States of America.

Ban, X., Gao, J., and van de Rijt, A. (2010). Navigation in Real-World Complex Networks through Embedding in Latent Spaces. In *Twelfth Workshop on Algorithm Engineering & Experiments*, pages 138–148, Austin, Texas - United States of America.

Barrière, L., Fraigniaud, P., Kranakis, E., and Krizanc, D. (2001). Efficient Routing in Networks with Long Range Contacts. In *15th International Symposium on Distributed Computing*, pages 270–284, Lisbon - Portugal.

Bollobás, B. and Chung, F. R. K. (1988). The Diameter of a Cycle Plus a Random Matching. *SIAM Journal on Discrete Mathematics*, 1(3):328–333.

Brady, A. and Cowen, L. (2006). Compact Routing on Power Law Graphs with Additive Stretch. In *Eighth Workshop on Algorithm Engineering & Experiments*, pages 119–128, Miami, Florida - United States of America.

Bringmann, K., Keusch, R., Lengler, J., Maus, Y., and Molla, A. R. (2017). Greedy Routing and the Algorithmic Small-World Phenomenon. In *ACM Symposium on Principles of Distributed Computing*, pages 371–380, Washington, District of Columbia - United States of America.

Chen, W., Sommer, C., Teng, S.-H., and Wang, Y. (2009). Compact Routing in Power-Law Graphs. In *23rd International Conference on Distributed Computing*, pages 379–391, Elche, Alicante - Spain.

Clarke, I., Sandberg, O., Wiley, B., and Hong, T. W. (2000). Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *ICSI Workshop on Design Issues in Anonymity and Unobservability*, pages 311–320, Berkeley, California - United States of America.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*, pages 658–664. The MIT Press, third edition.

Cowen, L. J. (1999). Compact Routing with Minimum Stretch. In *Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 255–260, Baltimore, Maryland - United States of America.

Dietzfelbinger, M. and Woelfel, P. (2009). Tight Lower Bounds for Greedy Routing in Uniform Small World Rings. In *Forty-first Annual ACM Symposium on Theory of Computing*, pages 591–600, Bethesda, Maryland - United States of America.

Erdős, P. (1963). Extremal problems in graph theory. In *Theory of Graphs and its Applications (Proc. Sympos. Smolenice)*, pages 29–36, Smolenice, Trnava - Slovakia.

Fraigniaud, P. and Gavoille, C. (2001). Routing in Trees. In *28th International Colloquium on Automata, Languages, and Programming*, pages 757–772, Hersonissos, Crete - Greece.

Fraigniaud, P. and Gavoille, C. (2002). A Space Lower Bound for Routing in Trees. In *19th Annual Symposium on Theoretical Aspects of Computer Science*, pages 65–75, Juan-les-Pins, Provence-Alpes-Côte d'Azur - France.

Fraigniaud, P., Gavoille, C., and Paul, C. (2003). Eclecticism Shrinks the World. Technical Report LRI-1376, University of Paris-Sud, Laboratory for Computer Science, Orsay, Île-de-France - France.

Fraigniaud, P., Gavoille, C., and Paul, C. (2006). Eclecticism shrinks even small worlds. *Distributed Computing*, 18(4):279–291.

Fredman, M. L., Komlós, J., and Szemerédi, E. (1984). Storing a Sparse Table with O(1) Worst Case Access Time. *Journal of the ACM*, 31(3):538–544.

Gavoille, C. and Gengler, M. (2001). Space-Efficiency for Routing Schemes of Stretch Factor Three. *Journal of Parallel and Distributed Computing*, 61(5):679–687.

Gavoille, C. and Hanusse, N. (1999). Compact Routing Tables for Graphs of Bounded Genus. In *26th International Colloquium on Automata, Languages and Programming*, pages 351–360, Prague, Bohemia - Czech Republic.

Heinonen, J. (2001). *Lectures on Analysis on Metric Spaces*, pages 1–9. Springer New York.

Kleinberg, J. (2000a). The Small-World Phenomenon: An Algorithmic Perspective. In *Thirty-second Annual ACM Symposium on Theory of Computing*, pages 163–170, Portland, Oregon - United States of America.

Kleinberg, J. (2001). Small-World Phenomena and the Dynamics of Information. In *14^{th} International Conference on Neural Information Processing Systems: Natural and Synthetic*, pages 431–438, Vancouver, British Columbia - Canada.

Kleinberg, J. (2006). Complex networks and decentralized search algorithms. In *International Congress of Mathematicians*, pages 1019–1044, Madrid - Spain.

Kleinberg, J. M. (2000b). Navigation in a small world. *Nature*, 406:845.

Konjevod, G., Richa, A. W., and Xia, D. (2006). Optimal-Stretch Name-Independent Compact Routing in Doubling Metrics. In *Twenty-fifth Annual ACM Symposium on Principles of Distributed Computing*, pages 198–207, Denver, Colorado - United States of America.

Konjevod, G., Richa, A. W., and Xia, D. (2007a). Compact Routing with Slack in Low Doubling Dimension. In *Twenty-sixth Annual ACM Symposium on Principles of Distributed Computing*, pages 71–80, Portland, Oregon - United States of America.

Konjevod, G., Richa, A. W., and Xia, D. (2007b). Optimal Scale-free Compact Routing Schemes in Networks of Low Doubling Dimension. In *Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 939–948, New Orleans, Louisiana - United States of America.

Krioukov, D., claffy, k., Fall, K., and Brady, A. (2007). On Compact Routing for the Internet. *Computer Communication Review*, 37(3):41–52.

Krioukov, D., Fall, K., and Yang, X. (2004). Compact Routing on Internet-Like Graphs. In *23^{rd} IEEE Conference on Computer Communications*, pages 209–219, Hong Kong, Pearl River - China.

Krioukov, D., Papadopoulos, F., Boguñá, M., and Vahdat, A. (2009). Greedy Forwarding in Scale-Free Networks Embedded in Hyperbolic Metric Spaces. *ACM SIGMETRICS Performance Evaluation Review*, 37(2):15–17.

Laing, K. and Rajaraman, R. (2007). A Space Lower Bound for Name-Independent Compact Routing in Trees. *Journal of Interconnection Networks*, 8(3):229–251.

Lebhar, E. and Schabanel, N. (2004). Almost Optimal Decentralized Routing in Long-Range Contact Networks. In *31^{st} International Colloquium on Automata, Languages, and Programming*, pages 894–905, Turku, Southwest Finland - Finland.

Liben-Nowell, D., Novak, J., Kumar, R., Raghavan, P., and Tomkins, A. (2005). Geographic routing in social networks. *Proceedings of the National Academy of Sciences*, 102(33):11623–11628.

Liu, C. and Wu, J. (2006). SWING: Small World Iterative Navigation Greedy Routing Protocol in MANETs. In *15^{th} International Conference on Computer Communications and Networks*, pages 339–350, Arlington, Virginia - United States of America.

Liu, X., Guan, J., Bai, G., and Lu, H. (2009). SWER: small world-based efficient routing for wireless sensor networks with mobile sinks. *Frontiers of Computer Science in China*, 3(3):427–434.

Lu, H.-I. (2002). Improved Compact Routing Tables for Planar Networks via Orderly Spanning Trees. In 8^{th} *International Computing and Combinatorics Conference*, pages 57–66, Singapore.

Manku, G. S., Bawa, M., and Raghavan, P. (2003). Symphony: Distributed Hashing In A Small World. In 4^{th} *USENIX Symposium on Internet Technologies and Systems*, pages 127–140, Seattle, Washington - United States of America.

Manku, G. S., Naor, M., and Wieder, U. (2004). Know thy Neighbor's Neighbor: the Power of Lookahead in Randomized P2P Networks. In *Thirty-sixth Annual ACM Symposium on Theory of Computing*, pages 54–63, Chicago, Illinois - United States of America.

Martel, C. and Nguyen, V. (2004). Analyzing Kleinberg's (and other) Small-world Models. In *Twenty-third Annual ACM Symposium on Principles of Distributed Computing*, pages 179–188, St. John's, Newfoundland and Labrador - Canada.

Milgram, S. (1967). The Small-World Problem. *Psychology Today*, 1(1):61–67.

Sandberg, O. (2006). Distributed Routing in Small-World Networks. In *Eighth Workshop on Algorithm Engineering & Experiments*, pages 144–155, Miami, Florida - United States of America.

Searcóid, M. Ó. (2007). *Metric Spaces*, pages 1–20. Springer London.

Shamim, M. S., Mansoor, N., Narde, R. S., Kothandapani, V., Ganguly, A., and Venkataraman, J. (2017). A Wireless Interconnection Framework for Seamless Inter and Intra-Chip Communication in Multichip Systems. *IEEE Transactions on Computers*, 66(3):389–402.

Strowes, S. D., Mooney, G., and Perkins, C. (2011). Compact Routing on the Internet AS-Graph. In 30^{th} *IEEE Conference on Computer Communications*, pages 852–857, Shanghai - China.

Talwar, K. (2004). Bypassing the Embedding: Approximation schemes and Compact Representations for Low Dimensional Metrics. In *Thirty-sixth Annual ACM Symposium on Theory of Computing*, pages 281–290, Chicago, Illinois - United States of America.

Tang, M., Yang, J., and Zhang, G. (2009). Compact Routing on Random Power Law Graphs. In *Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, pages 575–578, Chengdu, Sichuan - China.

Tang, M., Zhang, G., Lin, T., and Liu, J. (2013). HDLBR: A name-independent compact routing scheme for power-law networks. *Computer Communications*, 36(3):351–359.

Thorup, M. and Zwick, U. (2001). Compact routing schemes. In *Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 1–10, Heraklion, Crete - Greece.

Viertel, S. and Vignatti, A. L. (2015). Programação matemática e imersões métricas para aproximações em problemas de corte. *Revista de Informática Teórica e Aplicada*, 22(1):95–118.

Viertel, S. and Vignatti, A. L. (2018a). Labeling Algorithm and Compact Routing Scheme for a Small World Network Model. Technical Report arXiv:1806.01469, Federal University of Paraná, Curitiba, Paraná - Brazil.

Viertel, S. and Vignatti, A. L. (2018b). Small World Model based on a Sphere Homeomorphic Geometry. Technical Report arXiv:1808.01028, Federal University of Paraná, Curitiba, Paraná - Brazil.

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442.

Williamson, D. P. and Shmoys, D. B. (2010). *The Design of Approximation Algorithms*. Cambridge University Press.

Zeng, J., Hsu, W.-J., and Wang, J. (2005). Near Optimal Routing in a Small-World Network with Augmented Local Awareness. In *Third International Conference on Parallel and Distributed Processing and Applications*, pages 503–513, Nanjing, Jiangsu - China.

Zeng, J.-Y. and Hsu, W.-J. (2006). Optimal Routing in a Small-World Network. *Journal of Computer Science and Technology*, 21(4):476–481.

Zhang, H., Goel, A., and Govindan, R. (2002). Using the Small-World Model to Improve Freenet Performance. In *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 1228–1237, New York - United States of America.